Database "Trading Crypto"

Università degli Studi di Napoli Parthenope, Cds in Informatica

Attilio di Vicino, Mario Vista

A.A. 2021/2022

Indice

Elenco delle figure

Capitolo 1

Progettazione

Si vuole sviluppare un database per permettere ai trader di gestire le loro piattaforme di trading crypto

1.1 Requisiti

Il database permette ad un utente di gestire i propri conti virtuali su delle piattaforme di exchange di cryptovalute. Ogni utente effettua, in principio, una propria valutazione rischi, in cui decide in percentuale quanto è disposto a perdere del budget che investe. Tramite il database ogni utente può gestire sia i movimenti di deposito che di prelievo che vengono effettuati tra la carta di credito e il conto virtuale, sia gli scambi tra cryptovalute che vengono effettuati tramite l'exchange decentralizzato. Quest' ultimo fa riferimento al wallet decentralizzato da cui vengono prelevate per lo scambio le cryptovalute possedute dall'utente. Ogni utente può inserire diverse carte di credito, appartenenti a diverse banche, per ogni conto virtuale appartenente ad una sola piattaforma. L'utente inoltre riceve delle rewards in determinate date di rilascio corrispondenti a una determinata quantità di una cryptovaluta detta token che varia a seconda della Piattaforma in cui vengono rilasciate.

1.2 Glossario

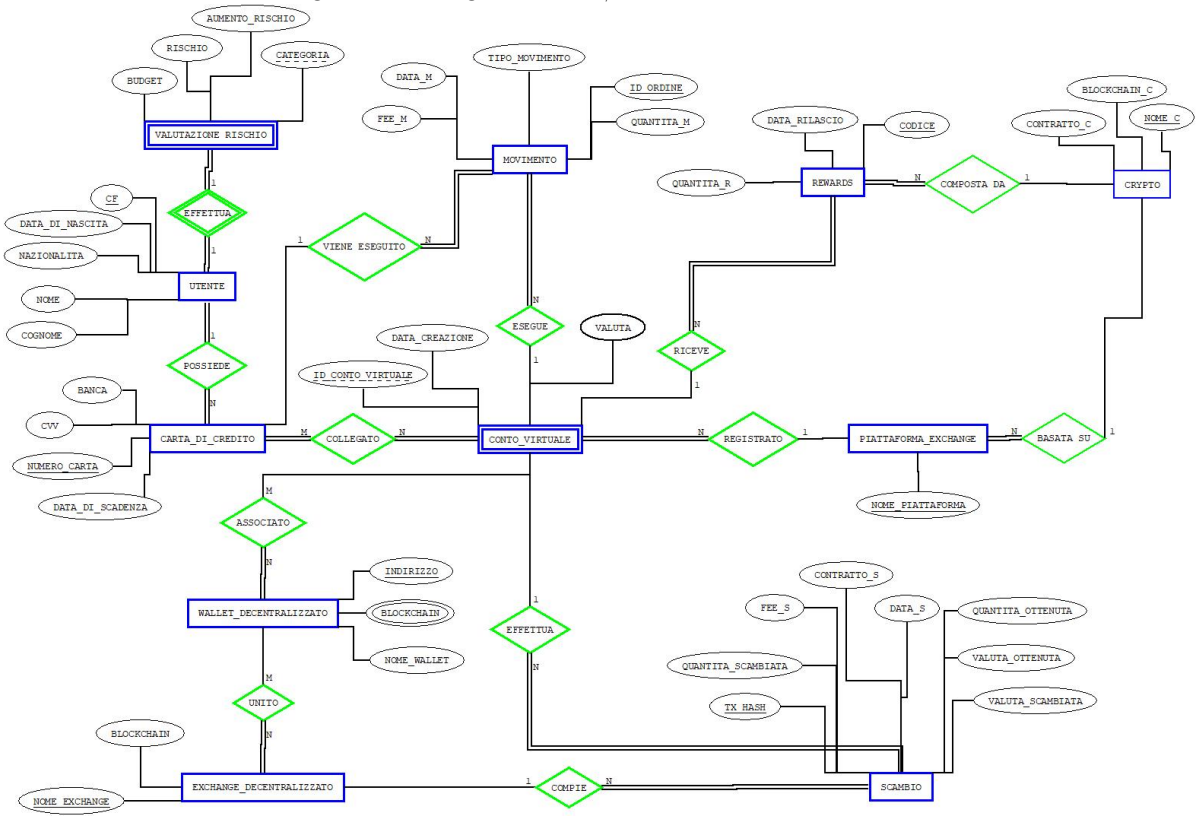
In questa sezione vengono chiariti i termini tecnici utilizzati nella sintesi di requisiti.

Glossario	
Termine	**Significato**
Cryptovalute	Rappresentazione digitale di valore basata sulla crittografia.
Blockchain	Rete di nodi che permette di gestire in modo univoco e sicuro le operazioni con le cryptovalute.
Piattaforma Exchange	Piattaforme possedute da società private che fanno da garante nelle operazioni di scambio.
Token di riferimento	Cryptovaluta di riferimento della piattaforma di exchange
Conto virtuale	Conto visualizzato all'interno della piattaforma di exchange.
Movimento	Movimento di deposito e prelievo che viene effettuato dalla carta di credito sul conto virtuale
Scambio	Scambio tra una valuta(euro,franchi o dollari) e una cryptovaluta, e viceversa, o tra cryptovalute
Valutazione rischi	Ogni utente determina, in percentuale, il rischio di perdita monetaria disposto a correre negli scambi.
Reward	Premi costituiti da una data quantità del token di riferimento della piattaforma che viene rilasciato al conto virtuale in determinate circostanze.
Wallet decentralizzato	Permette di utilizzare un Exchange decentralizzato, consente di visualizzare le cryptovalute possedute e di operare in forma anonima
Exchange Decentralizzato	Consente agli utenti di fare trading tra loro senza intermediari.
Contratto	Identificato da un codice, regolano l'accordo tra le parti che eseguono scambi le cryptovalute
Fee	Commissioni imposte su ogni operazione di scambio e su ogni movimento
Tx Hash	Abbreviazione di Transaction Hash, ossia una stringa unica di carattere attribuita ad ogni scambio verificato.

1.3 Diagramma EE/R

Nella figura 1.1 è possibile osservare il diagramma EE/R del database precedentemente descritto, con relative cardinalità e totalità. Non sono indicate le cardinalità massime e minime delle associazioni. Si noti che l'entità conto virtuale è debole poichè la sua esistenza dipende da quella della piattaforma di exchange.

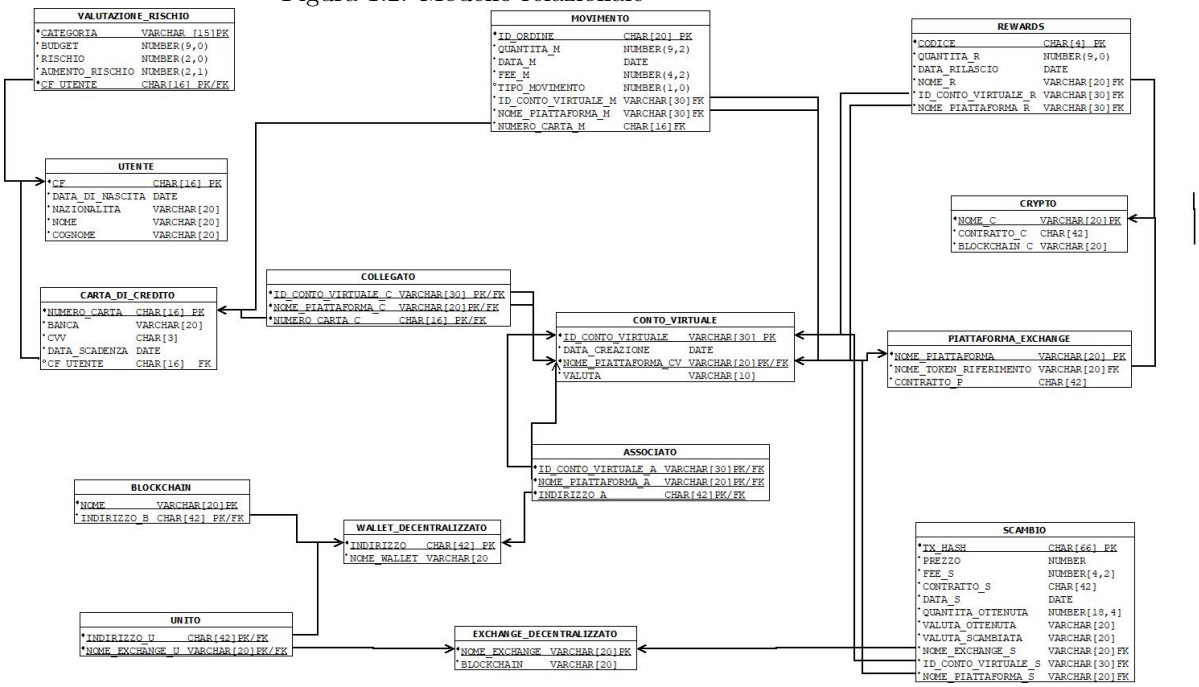
Figura 1.1: Diagramma EE/R



1.4 Modello Relazionale

Diagramma relazionale: Il diagramma relazionale è visibile in figura 1.2. Si noti che in movimento e scambio il tipo DATE comprende anche l'ora,sfruttando il tipo nativo date di oracle. Ricordando che l'entità Conto Virtuale è debole si noti che la chiave ID_CONTO_VIRTUALE da sola non è sufficiente, di fatti la chiave primaria è composta anche dal nome della piattaforma di exchange a cui è associata

Figura 1.2: Modello relazionale



1.5 Utenti e le loro categorie

1.5.1 Utenti

Le operazioni di trading avvengono tramite un unico utente le cui credenziali sono pubbliche, di conseguenza, nel database, avremo soltanto 3 utenti: un amministratore, un gestore del conto e il trader(account pubblico). Si fa notare che non sono indicati i privilegi di sistema, come il CONNECT, ma solo quelli di oggetto.

UTENTE	TIPO	VOLUME	PERMESSI
DB_TRADING	ADMIN	1	ALL
GESTORECONTO	COMUNE	1	SELECT,UPDATE,DELETE ON CONTO_VIRTUALE SELECT,UPDATE,DELETE ON COLLEGATO SELECT,UPDATE,DELETE ON ASSOCIATO SELECT ON MOVIMENTO SELECT ON REWARDS SELECT ON SCAMBIO SELECT ON CRYPTO SELECT ON ESTRATTO_CRYPTO EXECUTE ON ELIMINARE_CARTE_SCADUTE EXECUTE ON PREMIO EXECUTE ON PREMIO_SCAMBI
TRADER	COMUNE	1	SELECT,UPDATE ON VALUTAZIONE_RISCHIO SELECT ON UTENTE SELECT ON CRYPTO SELECT ON PIATTAFORMA_EXCHANGE SELECT ON CONTO_VIRTUALE SELECT ON COLLEGATO SELECT ON REWARDS SELECT ON WALLET_DECENTRALIZZATO SELECT ON BLOCKCHAIN SELECT ON ASSOCIATO SELECT ON EXCHANGE_DECENTRALIZZATO SELECT ON UNITO SELECT ON SCAMBIO SELECT ON ESTRATTO_CRYPTO UPDATE ON CARTA_DI_CREDITO EXECUTE ON AUMENTA_RISCHIO EXECUTE ON ESTRATTO

1.5.2 Operazione degli utenti

Le operazioni degli utenti più elementari sono quelle che possono essere eseguiti tramite comandi di DML, come ad esempio i movimenti o gli scambi effettuati dal Trader, per operazioni più complesse risulta però necessaria l'implementazione di stored procedures. Da notare che sia le operazioni DML che quelle che avvengono tramite procedure dovranno rispettare i vincoli d'integrità di cui discuteremo nella prossima sezione. Di seguito sono riportate le operazioni che ogni utente può compiere.

1. Il GESTORECONTO può eliminare le carte di credito scadute collegate al conto virtuale che hanno depositato meno di 500 euro(franchi o dollari) e collegate al conto da meno di un anno. Se nel conto virtuale non ci sono altre carte di credito il questo viene eliminato.
2. Il GESTORECONTO rilascia una reward al trader che ha versato di più su ogni singolo conto.
3. Il GESTORECONTO rilascia una reward sui conti che hanno compiuto almeno 10 scambi negli ultimi 30 giorni.
4. Il TRADER aumenta il rischio di un valore prefissato se almeno in un conto virtuale ha il saldo maggiore o uguale di 500 euro(franchi o dollari).
5. Il TRADER può visualizzare l'estratto conto dei suoi movimenti e dei suoi scambi.

Le seguenti tabelle mostrano le funzionalità delle procedure:

1.5.3 ELIMINA_CARTE_SCADUTE

OPERAZIONE	**ELIMINA_CARTE_SCADUTE**
SCOPO	Eliminare le carte scadute nelle condizioni citate sopra
ARGOMENTI	NULL
RISULTATI	Cancella CARTA_DI_CREDITOe, in caso CONTO_VIRTUALE
USA	COLLEGATO,CARTA_DI_CREDITO,CONTO_VIRTUALE
MODIFICA	CARTE_DI_CREDITO,COLLEGATO
PRIMA	Ci sono carte scadute nel database
POI	Non ci sono carte scadute nel database

1.5.4 PREMIO

OPERAZIONE	**PREMIO**
SCOPO	Premiare i trader che hanno versato più soldi
ARGOMENTI	NULL
RISULTATI	Rilascia reward su CONTO_VIRTUALE
USA	COLLEGATO,CARTA_DI_CREDITO,CONTO_VIRTUALE,UTENTE, REWARD,PIATTAFORMA_DI_EXCHANGE,MOVIMENTO
MODIFICA	REWARD

1.5.5 PREMIO_SCAMBI

OPERAZIONE	**PREMIO_SCAMBI**
SCOPO	Rilascia una reward se negli ultimi 30 giorni sono stati compiuti almeno 10 scambi
ARGOMENTI	NULL
RISULTATI	Rilascia una Reward
USA	COLLEGATO,CARTA_DI_CREDITO,CONTO_VIRTUALE,UTENTE, REWARD,PIATTAFORMA_DI_EXCHANGE,SCAMBIO
MODIFICA	REWARD

1.5.6 AUMENTA_RISCHIO

OPERAZIONE	**AUMENTA RISCHIO**
SCOPO	Aumentare percentuale di valutazione rischio
ARGOMENTI	CODICEFISCALE
RISULTATI	Rischio aumentato
USA	COLLEGATO,CARTA_DI_CREDITO,PIATTAFORMA_DI_EXCHANGE, VALUTAZIONE_RISCHIO
MODIFICA	VALUTAZIONE RISCHIO
PRIMA	La valutazione è quella inserita dal trader
POI	La valutazione rischio aumenta di quanto espresso dal Trader

1.5.7 ESTRATTO

OPERAZIONE	ESTRATTO
SCOPO	Visualizzare l'estratto conto
ARGOMENTI	CODFISCALE
RISULTATI	Visualizzazione estratto conto
USA	MOVIMENTO, CARTA_DI_CREDITO, SCAMBIO,COLLEGATO,

Di seguito è riportata la tabella in cui sono rappresentati i volumi delle procedure, ipotizzando che l'estratto conto venga visualizzato 10 volte al mese.

Nel TIPO la B sta per batch. Le operazioni sono batch poichè viene utilizzato sql statico.

OPERAZIONE	TIPO	VOLUME	PERIODO
ELIMINA_CARTE_SCADUTE	B	1	anno
PREMIO	B	1	mese
PREMIO_SCAMBI	B	1	mese
AUMENTA_RISCHIO	B	1	semestre
ESTRATTO	B	10	mese

1.6 Volumi

Di seguito viene riportata la tabella dei volumi per tutte le entità presenti nella base di dati. In questo caso ipotizziamo che nessun utente effettui la chiusura del conto da nessuna piattaforma di exchange e che nessun utente venga cancellato.

E sta per Entità, ED sta per enntità debole,A sta per tabelle di transizione

OPERAZIONE	TIPO	VOLUME	INCREMENTO	PERIODO
UTENTE	E	30	10	anno
VALUTAZIONE_RISCHIO	ED	30	10	anno
CARTA_DI_CREDITO	E	40	5	anno
CRYPTO	E	100	50	anno
PIATTAFORMA_EXCHANGE	E	20	5	anno
CONTO_VIRTUALE	E	20	5	anno
COLLEGATO	A	30	10	anno
MOVIMENTO	E	30	20	mese
REWARDS	E	10	5	mese
WALLET_DECENTRALIZZATO	E	25	5	anno
BLOCKCHAIN	E	5	1	semestre
ASSOCIATO	A	35	10	anno
EXCHANGE_DECENTRALIZZATO	E	25	5	anno
UNITO	A	35	10	anno
SCAMBIO	E	100	50	mese

1.7 Vincoli di integrità

I vincoli di integrità si suddividono in statici e dinamici:

- Statici: servono a controllare se i valori inseriti all'interno delle tabelle, indipendentemente dal tempo, siano ammessi.
- Dinamici: implementati tramite dei trigger, essi vengono automaticamente eseguiti all'avvenimento di un evento che può essere l'inserimento, la modifica o l'eliminazione di una tabella. Nel caso in cui i vincoli non siano rispettati verrà generato un messaggio d'errore.

Di seguito sono riportati i vincoli di integrità, fatta eccezione per quelli di chiave primaria e chiave esterna, in quanto ovvi.

1.7.1 Vincoli di integrità statici

- L'utente deve identificarsi in una tra le possibili categorie di trader: Intraday,Scalper,Buy & Hold.
- L'aumento del rischio dichiarato dal Trader deve essere tra lo 0.1% e il 2%.
- Il budget inserito deve essere positivo.
- Il rischio deve essere sempre positivo.
- Il CVV della carta di credito deve essere unico e non nullo.
- Le valute utilizzabili dal conto virtuale per l'acquisto di cryptovalute sono Euro, Dollari americani e Franchi svizzeri.
- I movimenti compiuti dall'utente devono essere di una quantità maggiore di zero.
- I movimenti possono essere solo di prelievo o di deposito, identificati rispettivamente da 1 e 2.
- La quantità delle reward deve essere positiva.

1.7.2 Vincoli di integrità dinamici

Questi vincoli, implementati tramite dei trigger che vengono automaticamente eseguiti all'avvenimento di un evento che può essere l'inserimento, la modifica o l'eliminazione di una tabella. Nel caso i vincoli non venissero rispettati verà generato un messaggio d'errore.

1. L'utente al momento della registrazione deve essere maggiorenne.
2. Quando viene inserita una carta viene controllato che essa non sia scaduta.

3. Quando viene effettuato un prelievo la quantità che si vuole prelevare deve essere disponibile sul saldo del conto.
4. La data della reward che viene rilasciata ad un conto virtuale deve essere successiva alla data di creazione di quest'ultimo.
5. La carta che viene collegata al conto virtuale deve essere esistente.
6. Viene controllato se tutte le carte collegate ad un conto virtuale appartengano allo stesso utente.
7. Quando viene effettuato un movimento la carta di credito che lo esegue deve essere collegata al rispettivo conto virtuale.
8. Quando viene effettuato uno scambio controlla che l'exchange decentralizzato sia collegato ad un wallet, a sua volta associato al conto virtuale che effettua lo scambio.
9. Prima di ogni scambio viene controllato se sul conto virtuale sono disponibili i fondi affinchè lo scambio avvenga.

1.8 Verifica di normalità

Nella verifica di normalità si cerca di azzerare, o diminuire il più possibile, le ridondanze. Ci sono 3 forme di normalizzazione, in particolare la prima risulta quasi naturale nel passaggio tra modello concettuale e relazionale. Di seguito vengono riportate tutte le normalizzazioni eseguite.

1.8.1 Prima forma normale

Nell'entità WALLET, l'attributo BLOCKCHAIN è multivalore, pertanto nella rappresentazione del modello relazionale, esso diventa una tabella a sè. Gli attributi ID_CONTO_VIRTUALE, INDIRIZZO,TX_HASH e CONTRATTO, rispettivamente delle entità ID_CONTO_VIRTUALE, WALLET_DECENTRALIZZATO, SCAMBIO, CRYPTO possono non sembrare operazioni atomiche poichè potrebbero essere suddivise in più campi, ma in questo caso lo sono poichè sono chiavi primarie. L'attributo CATEGORIA in VALUTAZIONE_RISCHIO potrebbe essere derivabile, ma in questo specifico caso non lo è poichè la scelta della categoria non è vincolante nell'uso delle operazioni messe a disposizione del trader, inoltre è chiave debole.

1.8.2 Seconda forma normale

Questa forma normale si applica nel caso in cui un'entità abbia due chiavi primarie e gli attributi dipendano solo da una delle due chiavi primarie. Il modello concettuale non presenta questa casistica dunque risulta essere in seconda forma normale.

1.8.3 Terza forma normale

Questa forma normale si applica nel caso in cui ci siano attributi non chiave che dipendono da altri attributi non chiave. In questo caso nel passaggio al modello relazionale bisogna rimuovere questi attributi dalla tabella dell'entità e inserirli in un'altra tabella. Il modello concettuale non presenta questa casistica dunque risulta essere in terza forma normale. Possiamo dunque aggiungere che la base di dati è anche nella forma normale di Boys e Codd.

1.8.4 Normalizzazione associazioni M a N

Di seguito vengono riportate le associazioni trasformate in entità essendo relazioni M a N:

- Collegato.
- Associato.
- Unito.

Capitolo 2

Implementazione

Nella fase di implementazione tutto ciò che è stato progettato in modello concettuale, successivamente tradotto in relazionale, viene trasformato in codice eseguibile. In particolare viene utilizzato il DBMS Oracle 11g-r2 XE e il linguaggio adottato è il PL/SQL. L'ordine del codice proposto non è obbligatorio ma segue logicamente gli argomenti trattati in fase di progettazione

2.1 Creazione utenti

Il primo passo da compiere è accedere al DBMS come amministratore e creare l'utente proprietario della base di dati. Possono essere creati anche altri utenti a cui però non possono essere dati i permessi poichè ancora non è ancora stato creato lo schema.

```
1 CREATE USER DB_TRADING IDENTIFIED BY ADMIN;
2 CREATE USER GESTORECONTO IDENTIFIED BY GESTORECONTO;
3 CREATE USER TRADER IDENTIFIED BY TRADER;
4
5 GRANT ALL PRIVILEGES TO DB_TRADING;
```

In seguito bisognerà uscire ed effettuare di nuovo l'accesso come DB_TRADING, ossia l'amministratore del DBMS che stiamo andando a creare, in modo da avere tutti i permessi che servono.

2.2 Data Definition Language

Il DDL è la trasformazione in codice dello schema relazionale. Le tabelle vengono prima interamente eliminate, in modo da evitare eventuali conflitti se il codice venisse avviato più volte, in seguito vengono inserite con il comando CREATE TABLE all'interno del quale, oltre a tutti gli elementi della rispettiva tabella del modello relazionale, vengono inclusi tutti i vincoli d'integrità statici come ad esempio l'obbligatorietà, i vincoli di dominio statici e l'integrità referenziale. Nelle tabelle che avranno la loro chiave primaria come chiave esterna in altre

tabelle bisogna forzare il loro DROP tramite il comando cascade constraints. Come è possibile notare dal modello concettuale, poichè tutte le tabelle che hanno chiavi esterne sono totali rispetto alle tabelle a cui queste ultime fanno riferimento, tutte le chiavi esterne presentano il vincolo NOT NULL.

2.2.1 Drop per evitare conflitti

```
DROP TABLE UTENTE cascade constraints;
DROP TABLE VALUTAZIONE_RISCHIO;
DROP TABLE CARTA_DI_CREDITO cascade constraints;
DROP TABLE CRYPTO cascade constraints;
DROP TABLE PIATTAFORMA_EXCHANGE cascade constraints;
DROP TABLE CONTO_VIRTUALE cascade constraints;
DROP TABLE COLLEGATO;
DROP TABLE MOVIMENTO;
DROP TABLE REWARDS;
DROP TABLE WALLET_DECENTRALIZZATO cascade constraints;
DROP TABLE BLOCKCHAIN;
DROP TABLE ASSOCIATO;
DROP TABLE EXCHANGE_DECENTRALIZZATO cascade constraints;
DROP TABLE UNITO;
DROP TABLE SCAMBIO;
```

2.2.2 Utente

Il popolamento avviene direttamente tramite la registrazione da parte dei Trader.

```
CREATE TABLE UTENTE(
    CF              CHAR(16)     NOT NULL PRIMARY KEY,
    DATA_DI_NASCITA DATE         NOT NULL,
    NAZIONALITA     VARCHAR(20) NOT NULL,
    NOME            VARCHAR(20) NOT NULL,
    COGNOME         VARCHAR(20) NOT NULL
);
```

2.2.3 Valutazione rischio

Il popolamento di questa tabella avviene nel momento in cui i Trader inseriscono la loro valutazione rischi.Utilizziamo dei constraint per far si che la scelta della categoria e l'aumento del rischio siano valori ammessi dal database.

```
CREATE TABLE VALUTAZIONE_RISCHIO(
    CATEGORIA       VARCHAR(15) NOT NULL,
    BUDGET          NUMBER(9,0) NOT NULL,
    RISCHIO         NUMBER(2,0) NOT NULL,
    CF_UTENTE       CHAR(16)    NOT NULL,
    AUMENTO_RISCHIO NUMBER(2,1) NOT NULL,

    CONSTRAINT  PK_VALUTAZIONE_RISCHIO PRIMARY KEY(CATEGORIA,
    CF_UTENTE),
    CONSTRAINT  FK_UTENTE FOREIGN KEY (CF_UTENTE) REFERENCES UTENTE
    (CF) ON DELETE CASCADE,

```

```
    CONSTRAINT  CATEGORIE_AMMESSE CHECK ( LOWER (CATEGORIA) IN ('intraday','scalper','buy & hold') ),
    CONSTRAINT  COMP_AUMENTO_RISCHIO CHECK (AUMENTO_RISCHIO BETWEEN 0.1 AND 2.0),
    CONSTRAINT  BUDGET_POS CHECK (BUDGET > 0),
    CONSTRAINT  RISCHIO_POS CHECK (RISCHIO > 0)
);
```

2.2.4 Carta di credito

È costituito dall'insieme dei metodi di pagamento inseriti dai Trader per compiere operazioni di deposito prelievo.

```
CREATE TABLE CARTA_DI_CREDITO(
    NUMERO_CARTA        CHAR(16)     NOT NULL PRIMARY KEY,
    CVV                 CHAR(3)      NOT NULL UNIQUE,
    DATA_DI_SCADENZA    DATE         NOT NULL,
    BANCA               VARCHAR(20)  NOT NULL,
    CF_UTENTE           CHAR(16)     NOT NULL,

    CONSTRAINT  FK_UTENTE2 FOREIGN KEY (CF_UTENTE) REFERENCES
    UTENTE(CF) ON DELETE CASCADE
);
```

2.2.5 Crypto

Fondamentale poichè ci indica la cryptovaluta su cui opera la piattaforma di exchange, inoltre ci permette di conoscere la cryptovaluta ricevuta tramite reward

```
CREATE TABLE CRYPTO(
    NOME_C          VARCHAR(20)  NOT NULL PRIMARY KEY,
    CONTRATTO_C     CHAR(42)     NOT NULL,
    BLOCKCHAIN_C    VARCHAR(20)  NOT NULL
);
```

2.2.6 Piattaforma Exchange

Ci consente di conoscere la piattaforma sulla quale l'utente ha il conto virtuale e che opera come garante degli scambi e dei movimenti effetuati dal conto stesso.

```
CREATE TABLE PIATTAFORMA_EXCHANGE(
    NOME_PIATTAFORMA    VARCHAR(20)     NOT NULL PRIMARY KEY,
    TOKEN_RIFERIMENTO   VARCHAR(20)     NOT NULL,

    CONSTRAINT FK_TOKEN_RIFERIMENTO FOREIGN KEY (TOKEN_RIFERIMENTO)
     REFERENCES CRYPTO(NOME_C) ON DELETE CASCADE
);
```

2.2.7 Conto virtuale

Conto su cui sono presenti i soldi con cui possono essere compiuti scambi o movimenti. La sua esistenza dipende dalla piattaforma su cui è registrato. Le valute utilizzabili sono euro franco svizzero e dollaro statunitense ma in realtà è possibile estenderle anche con altre valute.

```
CREATE TABLE CONTO_VIRTUALE(
    ID_CONTO_VIRTUALE   VARCHAR(30) NOT NULL,
    NOME_PIATTAFORMA_CV VARCHAR(20) NOT NULL,
    DATA_CREAZIONE      DATE        NOT NULL,
    VALUTA              VARCHAR(10) NOT NULL,

    CONSTRAINT PK_CONTO_VIRTUALE PRIMARY KEY (ID_CONTO_VIRTUALE,
    NOME_PIATTAFORMA_CV),
    CONSTRAINT FK_PIATTAFORMA FOREIGN KEY (NOME_PIATTAFORMA_CV)
    REFERENCES PIATTAFORMA_EXCHANGE(NOME_PIATTAFORMA) ON DELETE
    CASCADE,
    CONSTRAINT VALUTE_DISPONIBILI CHECK (LOWER (VALUTA) IN ('eur','
    usd','chf'))
);
```

2.2.8 Collegato

Tabella di transizione che unisce il conto virtuale alla carta di credito.

```
CREATE TABLE COLLEGATO(
    ID_CONTO_VIRTUALE_C   VARCHAR(30)   NOT NULL,
    NOME_PIATTAFORMA_C    VARCHAR(20)   NOT NULL,
    NUMERO_CARTA_C        CHAR(16)      NOT NULL,

    CONSTRAINT  PK_COLLEGATO PRIMARY KEY (ID_CONTO_VIRTUALE_C,
    NUMERO_CARTA_C,NOME_PIATTAFORMA_C),
    CONSTRAINT  FK_ID FOREIGN KEY (ID_CONTO_VIRTUALE_C,
    NOME_PIATTAFORMA_C) REFERENCES CONTO_VIRTUALE(ID_CONTO_VIRTUALE
    ,NOME_PIATTAFORMA_CV) ON DELETE CASCADE,
    CONSTRAINT  FK_NUMERO_CONTO FOREIGN KEY (NUMERO_CARTA_C)
    REFERENCES CARTA_DI_CREDITO(NUMERO_CARTA) ON DELETE CASCADE
);
```

2.2.9 Movimento

Il movimento consente all'utente il passaggio di denaro di una valuta nota tramite depositi o prelievi che avvengono tra la carta di credito e il conto virtuale.

```
CREATE TABLE MOVIMENTO(
    ID_ORDINE               CHAR(20)        NOT NULL PRIMARY KEY,
    QUANTITA_M              NUMBER(9,2)     NOT NULL,
    DATA_M                  DATE            NOT NULL, -- NON E'
    UNIQUE
    FEE_M                   NUMBER(4,2)     NOT NULL, -- INUTILE
    CHECK
    TIPO_MOVIMENTO          NUMBER(2,0)     NOT NULL,
```

```
    ID_CONTO_VIRTUALE_M     VARCHAR(30)     NOT NULL,
    NUMERO_CARTA_M          CHAR(16)        NOT NULL,
    NOME_PIATTAFORMA_M      VARCHAR(30)     NOT NULL,

    CONSTRAINT  FK_ID_CONTO_VIRTUALE FOREIGN KEY (
    ID_CONTO_VIRTUALE_M,NOME_PIATTAFORMA_M) REFERENCES
    CONTO_VIRTUALE(ID_CONTO_VIRTUALE,NOME_PIATTAFORMA_CV) ON DELETE
     CASCADE,
    CONSTRAINT  FK_CARTA_DI_CREDITO FOREIGN KEY (NUMERO_CARTA_M)
    REFERENCES CARTA_DI_CREDITO(NUMERO_CARTA) ON DELETE CASCADE,

    CONSTRAINT  MOVIMENTO_CONSENTITO CHECK ( QUANTITA_M > 0 ),
    CONSTRAINT  TIPO_MOVIMENTO_CONSENTITO CHECK ( (TIPO_MOVIMENTO =
     1) OR (TIPO_MOVIMENTO = 2) )
);
```

2.2.10 Rewards

Premi in cryptovalute note che vengono rilasciati all'utente in determinate condizioni tramite procedure. Il popolamento di questa tabella è automatico

```
CREATE TABLE REWARDS(
    CODICE              NUMBER          GENERATED BY DEFAULT AS
    IDENTITY PRIMARY KEY,
    QUANTITA_R          NUMBER(9,0)     NOT NULL,
    DATA_RILASCIO       DATE            NOT NULL,
    NOME_R              VARCHAR(20)     NOT NULL,
    ID_CONTO_VIRTUALE_R VARCHAR(30)     NOT NULL,
    NOME_PIATTAFORMA_R  VARCHAR(20)     NOT NULL,

    CONSTRAINT  FK_NOME_R FOREIGN KEY (NOME_R) REFERENCES CRYPTO (
    NOME_C) ON DELETE CASCADE,
    CONSTRAINT  FK_ID_CONTO_VIRTUALE_R FOREIGN KEY (
    ID_CONTO_VIRTUALE_R,NOME_PIATTAFORMA_R) REFERENCES
    CONTO_VIRTUALE(ID_CONTO_VIRTUALE,NOME_PIATTAFORMA_CV) ON DELETE
     CASCADE,
    CONSTRAINT  QUANTITA_POS CHECK (QUANTITA_R > 0)
);
```

2.2.11 Wallet decentralizzato

Mediatore per l'utilizzo dell'exchange decentralizzato.

```
CREATE TABLE WALLET_DECENTRALIZZATO(
    INDIRIZZO    CHAR(42)    NOT NULL PRIMARY KEY,
    NOME_WALLET  VARCHAR(20) NOT NULL
);
```

2.2.12 Blockchain

Necessaria per sapere il circuito di cryptovalute su cui opera il wallet.

```
CREATE TABLE BLOCKCHAIN(
    NOME        CHAR(29)    NOT NULL,
```

```
    INDIRIZZO_B CHAR(42)    NOT NULL,

    CONSTRAINT PK_BLOCKCHAIN PRIMARY KEY (NOME,INDIRIZZO_B),
    CONSTRAINT FK_BLOCKCHAIN FOREIGN KEY (INDIRIZZO_B) REFERENCES
    WALLET_DECENTRALIZZATO(INDIRIZZO) ON DELETE CASCADE
);
```

2.2.13 Associato

Tabella di transizione che collega il conto virtuale al wallet decentralizzato.

```
CREATE TABLE ASSOCIATO(
    ID_CONTO_VIRTUALE_A   VARCHAR(30)   NOT NULL,
    NOME_PIATTAFORMA_A    VARCHAR(20)   NOT NULL,
    INDIRIZZO_A           CHAR(42)      NOT NULL,

    CONSTRAINT  PK_ASSOCIATO PRIMARY KEY (ID_CONTO_VIRTUALE_A,
    NOME_PIATTAFORMA_A,INDIRIZZO_A),
    CONSTRAINT  FK_CV FOREIGN KEY (ID_CONTO_VIRTUALE_A,
    NOME_PIATTAFORMA_A) REFERENCES CONTO_VIRTUALE(ID_CONTO_VIRTUALE
    ,NOME_PIATTAFORMA_CV) ON DELETE CASCADE,
    CONSTRAINT  FK_WALLET FOREIGN KEY (INDIRIZZO_A) REFERENCES
    WALLET_DECENTRALIZZATO(INDIRIZZO) ON DELETE CASCADE
);
```

2.2.14 Exchange decentralizzato

Permette di effettuare gli scambi di cryptovalute.

```
CREATE TABLE EXCHANGE_DECENTRALIZZATO(
    NOME_EXCHANGE    VARCHAR(20)    NOT NULL PRIMARY KEY,
    BLOCKCHAIN       VARCHAR(20)    NOT NULL
);
```

2.2.15 Unito

Tabella di transizione che collega il wallet decentralizzato e l'exchange decentralizzato.

```
CREATE TABLE UNITO(
    INDIRIZZO_U     CHAR(42)    NOT NULL,
    NOME_EXCHANGE_U VARCHAR(20) NOT NULL,

    CONSTRAINT PK_UNITO PRIMARY KEY(INDIRIZZO_U,NOME_EXCHANGE_U),
    CONSTRAINT FK_WALLET_U FOREIGN KEY (INDIRIZZO_U) REFERENCES
    WALLET_DECENTRALIZZATO(INDIRIZZO) ON DELETE CASCADE,
    CONSTRAINT FK_DEX FOREIGN KEY (NOME_EXCHANGE_U) REFERENCES
    EXCHANGE_DECENTRALIZZATO(NOME_EXCHANGE) ON DELETE CASCADE
);
```

2.2.16 Scambio

Operazione di Trading vera e propria. Può avvenure tra due cryptovalute, tra una valuta e una cryptovaluta e viceversa

```
CREATE TABLE SCAMBIO(
    TX_HASH              CHAR(66)         NOT NULL PRIMARY KEY,
    QUANTITA_SCAMBIATA   NUMBER(18,4)     NOT NULL,
    FEE_S                NUMBER(4,2)      NOT NULL,
    CONTRATTO_S          CHAR(42)         NOT NULL,
    DATA_S               DATE             NOT NULL, -- NON E' UNIQUE
    QUANTITA_OTTENUTA    NUMBER(18,4)     NOT NULL,
    VALUTA_OTTENUTA      VARCHAR(20)      NOT NULL,
    VALUTA_SCAMBIATA     VARCHAR(20)      NOT NULL,
    ID_CONTO_VIRTUALE_S  VARCHAR(30)      NOT NULL,
    NOME_PIATTAFORMA_S   VARCHAR(30)      NOT NULL,
    NOME_EXCHANGE_DEX    VARCHAR(20)      NOT NULL,

    CONSTRAINT FK_ID_CONTO_VIRTUALE_S FOREIGN KEY (
    ID_CONTO_VIRTUALE_S,NOME_PIATTAFORMA_S) REFERENCES
    CONTO_VIRTUALE(ID_CONTO_VIRTUALE,NOME_PIATTAFORMA_CV) ON DELETE
     CASCADE,
    CONSTRAINT FK_DEX_S FOREIGN KEY (NOME_EXCHANGE_DEX) REFERENCES
    EXCHANGE_DECENTRALIZZATO(NOME_EXCHANGE) ON DELETE CASCADE
);
```

2.3 Data Manipulation Language

Il popolamento di tutte le tabelle avviene tramite i comandi messi a disposizione dal linguaggio SQL, ossia la INSERT o l'UPDATE. L'inserimento delle tuple su tutte le tabelle è affidato all'admin. Di seguito viene proposta una tupla d'esempio per ogni tabella.

```
INSERT INTO UTENTE(CF,DATA_DI_NASCITA,NAZIONALITA,NOME,COGNOME)
    VALUES
('DSRLRD90A01F839V',TO_DATE('01/06/1999','DD/MM/YYYY'),'ITALIA','
    ALFREDO','DESERTO');

INSERT INTO VALUTAZIONE_RISCHIO(CATEGORIA,BUDGET,RISCHIO,CF_UTENTE,
    AUMENTO_RISCHIO) VALUES
('INTRADAY',10000,5,'DSRLRD90A01F839V',1);

INSERT INTO CARTA_DI_CREDITO(NUMERO_CARTA,CVV,DATA_DI_SCADENZA,
    CF_UTENTE,BANCA) VALUES
('4109996351112611','679',TO_DATE('01/06/2023','DD/MM/YYYY'),'
    DSRLRD90A01F839V','UNICREDIT');

INSERT INTO CRYPTO(NOME_C,CONTRATTO_C,BLOCKCHAIN_C) VALUES
('BNB','0xKaP7e0yhG0X0GHwZJnv7d3Oe0S0DGj6uovM4MDpL','BSC');

INSERT INTO PIATTAFORMA_EXCHANGE(NOME_PIATTAFORMA,TOKEN_RIFERIMENTO
    ) VALUES
('BINANCE','BNB');

```

```
16 INSERT INTO CONTO_VIRTUALE(ID_CONTO_VIRTUALE,NOME_PIATTAFORMA_CV,
       DATA_CREAZIONE,VALUTA) VALUES
17 ('11013563144710829511','BITPANDA',TO_DATE('19/09/2010','DD/MM/YYYY
       '),'EUR');
18
19 INSERT INTO COLLEGATO(ID_CONTO_VIRTUALE_C,NUMERO_CARTA_C,
       NOME_PIATTAFORMA_C) VALUES
20 ('11013563144710829511','4109996351112611','BITPANDA');
21
22 INSERT INTO MOVIMENTO(ID_ORDINE,QUANTITA_M,DATA_M,FEE_M,
       TIPO_MOVIMENTO,ID_CONTO_VIRTUALE_M,NUMERO_CARTA_M,
       NOME_PIATTAFORMA_M) VALUES
23 ('57060783237376271012',150,TO_DATE('01/06/2022 14:29','DD/MM/YYYY
       HH24:MI'),1,1,'11013563144710829511','4109996351112611','
       BITPANDA');
24
25 INSERT INTO REWARDS(QUANTITA_R,DATA_RILASCIO,NOME_R,
       NOME_PIATTAFORMA_R,ID_CONTO_VIRTUALE_R) VALUES
26 (10,TO_DATE('01/08/2022','DD/MM/YYYY'),'BNB','BITPANDA','
       11013563144710829511');
27
28 INSERT INTO WALLET_DECENTRALIZZATO(INDIRIZZO,NOME_WALLET) VALUES
29 ('0xlmt9ivxtbwyz1ea6bbxxfc1vve5f5ni82i69d55y','METAMASK');
30
31 INSERT INTO BLOCKCHAIN(INDIRIZZO_B,NOME) VALUES
32 ('0xlmt9ivxtbwyz1ea6bbxxfc1vve5f5ni82i69d55y','BSC');
33
34 INSERT INTO ASSOCIATO(ID_CONTO_VIRTUALE_A,NOME_PIATTAFORMA_A,
       INDIRIZZO_A) VALUES
35 ('11013563144710829511','BITPANDA','0
       xlmt9ivxtbwyz1ea6bbxxfc1vve5f5ni82i69d55y');
36
37 INSERT INTO EXCHANGE_DECENTRALIZZATO(NOME_EXCHANGE,BLOCKCHAIN)
       VALUES
38 ('pancakeswap','BSC');
39
40 INSERT INTO UNITO(INDIRIZZO_U,NOME_EXCHANGE_U) VALUES
41 ('0xlmt9ivxtbwyz1ea6bbxxfc1vve5f5ni82i69d55y','pancakeswap');
42
43 INSERT INTO SCAMBIO(TX_HASH,QUANTITA_SCAMBIATA,FEE_S,CONTRATTO_S,
       DATA_S,QUANTITA_OTTENUTA,VALUTA_OTTENUTA,VALUTA_SCAMBIATA,
       ID_CONTO_VIRTUALE_S,NOME_PIATTAFORMA_S,NOME_EXCHANGE_DEX)
       VALUES
44 ('0
       xzibx1k4mlts22wlfbe5202beqho4nzmoyzd1nvh8r0wsg32i8vk5j7tllun2morb
       ',100,1,'0xKaP7e0yhG0X0GHwZJnv7d3Oe0S0DGj6uovM4MDpL',TO_DATE('
       10/07/2022','DD/MM/YYYY'),10,'BNB','EUR','11013563144710829511'
       ,'BITPANDA','pancakeswap');
45
46 COMMIT;
```

2.4 Trigger

I trigger DML sono molto utili per controllare i vincoli d'integrità dei dati inseriti all'avvenimento di un evento che può essere di inserimento, modifica o

eliminazione di una tupla.

2.4.1 VERIFICA_ETA_UTENTE

Questo trigger controlla, quando viene inserito un nuovo utente trader all'interno del database, che esso sia maggiorenne, poichè legalmente è possibile aprire conti di trading solo al compimento della maggiore età. Se il vincolo non è rispettato il trigger genera un messaggio d'errore

```
CREATE OR REPLACE TRIGGER VERIFICA_ETA_UTENTE
BEFORE INSERT ON UTENTE
FOR EACH ROW
DECLARE
    MINORENNE EXCEPTION;
BEGIN

    IF FLOOR((SYSDATE - :NEW.DATA_DI_NASCITA) / 365.25) < 18 THEN
        RAISE MINORENNE;
    END IF;

EXCEPTION WHEN MINORENNE THEN
    RAISE_APPLICATION_ERROR(-1000,'ERRORE, DEVI AVERE ALMENO 18
    ANNI!');

END VERIFICA_ETA_UTENTE;
```

2.4.2 VERIFICA_CARTA_SCADUTA

Questo vincolo controlla che quando viene inserita o aggiornata una nuova carta di credito, essa non sia scaduta. Se il vincolo non è rispettato il trigger genera un messaggio d'errore. Si noti che dopo l'update di una tabella andrebbe fatto il commit, ossia il salvataggio permanente dei dati del database, ma in questo caso trattandosi di un trigger non va inserito.

```
CREATE OR REPLACE TRIGGER VERIFICA_CARTA_SCADUTA
BEFORE INSERT OR UPDATE ON CARTA_DI_CREDITO
FOR EACH ROW
DECLARE
    CARTASCADUTA EXCEPTION;
BEGIN

    IF :NEW.DATA_DI_SCADENZA < SYSDATE THEN
        RAISE CARTASCADUTA;
    END IF;

EXCEPTION WHEN CARTASCADUTA THEN
    RAISE_APPLICATION_ERROR(-2000,'ERRORE, CARTA SCADUTA!');

END VERIFICA_CARTA_SCADUTA;
```

2.4.3 CONTROLLO_PRELIEVO

Questo trigger controlla che, quando viene effettuato un prelievo, i soldi che vogliono essere prelevati siano effettivamente disponibili sul conto virtuale. Per prima cosa viene controllato se il conto virtuale ha effettuato almeno un movimento, in caso contrario genera un messaggio d'errore. In seguito, se sono stati effettuati movimenti, viene richiamata la function SALDOVAL,che ritorna il saldo, che viene confrontato con la quantità che si vuole prelevare.

```
CREATE OR REPLACE TRIGGER CONTROLLO_PRELIEVO
BEFORE INSERT ON MOVIMENTO
FOR EACH ROW
DECLARE
    SALDO_NON_SUFFICIENTE   EXCEPTION;
    SALDO   NUMBER := 0;
    VALUTA_ VARCHAR(20);
    CONTATORE NUMBER := 0;
BEGIN

    SELECT COUNT(*)
    INTO CONTATORE
    FROM MOVIMENTO
    WHERE ID_CONTO_VIRTUALE_M = :NEW.ID_CONTO_VIRTUALE_M AND
    NOME_PIATTAFORMA_M = :NEW.NOME_PIATTAFORMA_M;

    IF CONTATORE > 0 AND :NEW.TIPO_MOVIMENTO = 2 THEN

        SELECT VALUTA
        INTO VALUTA_
        FROM CONTO_VIRTUALE
        WHERE ID_CONTO_VIRTUALE = :NEW.ID_CONTO_VIRTUALE_M AND
    NOME_PIATTAFORMA_CV = :NEW.NOME_PIATTAFORMA_M;

        SALDO := SALDOVAL(:NEW.ID_CONTO_VIRTUALE_M,:NEW.
    NOME_PIATTAFORMA_M,VALUTA_);

        IF :NEW.QUANTITA_M > SALDO THEN
            RAISE SALDO_NON_SUFFICIENTE;
        END IF;
    -- NEL CASO IN CUI NON CI SIA NESSUNA TUPLA IN MOVIMENTO E SI
    VUOLE PRELEVARE
    ELSIF :NEW.TIPO_MOVIMENTO = 2 THEN
        RAISE SALDO_NON_SUFFICIENTE;

    END IF;

    EXCEPTION WHEN SALDO_NON_SUFFICIENTE THEN
        RAISE_APPLICATION_ERROR(-3000,'ERRORE, SALDO INSUFFICIENTE!
    ');

END CONTROLLO_PRELIEVO;
```

2.4.4 CONTROLLO_DATA_REWARDS

Viene effettuato un controllo prima del rilascio delle reward per cui la data di rilascio feve essere antecedente alla data di creazione del conto virtuale

```
CREATE OR REPLACE TRIGGER CONTROLLO_DATA_REWARDS
BEFORE INSERT ON REWARDS
FOR EACH ROW
DECLARE
    DATA_NON_ACCETTATA EXCEPTION;
    DATACREAZIONE      DATE;
BEGIN

    SELECT DATA_CREAZIONE
    INTO DATACREAZIONE
    FROM CONTO_VIRTUALE
    WHERE ID_CONTO_VIRTUALE = :NEW.ID_CONTO_VIRTUALE_R AND
    NOME_PIATTAFORMA_CV = :NEW.NOME_PIATTAFORMA_R;

    IF DATACREAZIONE != NULL THEN
        IF :NEW.DATA_RILASCIO < DATACREAZIONE THEN
            RAISE   DATA_NON_ACCETTATA;
        END IF;
    END IF;

    EXCEPTION WHEN DATA_NON_ACCETTATA THEN
        RAISE_APPLICATION_ERROR(-4000,'ERRORE, IL CONTO NON ESISTE!
    ');
END CONTROLLO_DATA_REWARDS;
```

2.4.5 CONTROLLO_ESISTENZA_CARTA

Controlla che la carta di credito collegata al conto virtuale sia presente nel database.

```
CREATE OR REPLACE TRIGGER CONTROLLO_ESISTENZA_CARTA
BEFORE INSERT ON COLLEGATO
FOR EACH ROW
DECLARE
    NON_ESISTE  EXCEPTION;
    NCARTA      CHAR(16);
BEGIN

    SELECT NUMERO_CARTA
    INTO NCARTA
    FROM CARTA_DI_CREDITO
    WHERE NUMERO_CARTA = :NEW.NUMERO_CARTA_C;

    IF NCARTA IS NULL THEN
        RAISE NON_ESISTE;
    END IF;

    EXCEPTION WHEN NON_ESISTE THEN
        RAISE_APPLICATION_ERROR(-5000,'ERRORE, CARTA INESISTENTE!')
    ;

END CONTROLLO_ESISTENZA_CARTA;
```

2.4.6 VERIFICA_CARTE_SU_CONTO_VIRTUALE

Questo trigger,eseguito prima dell'inserimento della tupla, controlla se tutte le carte virtuali collegate ad un conto virtuale appartengano tutte allo stesso utente

```
CREATE OR REPLACE TRIGGER VERIFICA_CARTE_SU_CONTO_VIRTUALE
BEFORE INSERT ON COLLEGATO
FOR EACH ROW
DECLARE
    PIU_UTENTI  EXCEPTION;
    NCARTA      CHAR(16);
    CFUTENTE    CHAR(16);
BEGIN

    SELECT CF_UTENTE
    INTO CFUTENTE
    FROM CARTA_DI_CREDITO
    WHERE NUMERO_CARTA = :NEW.NUMERO_CARTA_C;

    FOR i IN (
        SELECT CF_UTENTE
            FROM CARTA_DI_CREDITO CC JOIN COLLEGATO C2 ON CC.
    NUMERO_CARTA = C2.NUMERO_CARTA_C
                WHERE C2.ID_CONTO_VIRTUALE_C = :NEW.
    ID_CONTO_VIRTUALE_C AND C2.NOME_PIATTAFORMA_C = :NEW.
    NOME_PIATTAFORMA_C
    )
    LOOP
        IF CFUTENTE != i.CF_UTENTE THEN
            RAISE PIU_UTENTI;
        END IF;
    END LOOP;

    EXCEPTION WHEN PIU_UTENTI THEN
        RAISE_APPLICATION_ERROR(-6000,'ERRORE, CARTE COLLEGATE A
    PIU DI UNA PERSONA!');

END VERIFICA_CARTE_SU_CONTO_VIRTUALE;
```

2.4.7 CONTROLLO_METODO_DI_PAGAMENTO

Prima del compimento di un movimento questo trigger controlla se la carta di credito è collegata al rispettivo conto virtuale.

```
CREATE OR REPLACE TRIGGER CONTROLLO_METODO_DI_PAGAMENTO
BEFORE INSERT ON MOVIMENTO
FOR EACH ROW
DECLARE
    NON_PRESENTE EXCEPTION;
    CONTAT NUMBER := 0;
BEGIN

    SELECT COUNT(NUMERO_CARTA_C)
    INTO CONTAT
    FROM COLLEGATO
```

```
12      WHERE :NEW.ID_CONTO_VIRTUALE_M = ID_CONTO_VIRTUALE_C AND
        NOME_PIATTAFORMA_C = :NEW.NOME_PIATTAFORMA_M AND :NEW.
        NUMERO_CARTA_M = NUMERO_CARTA_C;
13
14      IF CONTAT < 1 THEN
15          RAISE NON_PRESENTE;
16      END IF;
17
18      EXCEPTION WHEN NON_PRESENTE THEN
19          RAISE_APPLICATION_ERROR(-7000,'IL METODO DI PAGAMENTO NON
        RISULTA COLLEGATO AL CONTO VIRTUALE');
20
21  END CONTROLLO_METODO_DI_PAGAMENTO;
```

2.4.8 CONTROLLO_SCAMBIO

Prima dell'avvenimento di uno scambio viene controllato se l'exchange decentralizzato sia collegato ad un wallet decentralizzato associato a sua volta al conto virtuale. Questo è indispensabile affinchè lo scambio possa avvenire correttamente.

```
1   CREATE OR REPLACE TRIGGER CONTROLLO_SCAMBIO
2   BEFORE INSERT ON SCAMBIO
3   FOR EACH ROW
4   DECLARE
5       NON_COLLEGATO EXCEPTION;
6       CONTA NUMBER := 0;
7   BEGIN
8
9       SELECT COUNT(*)
10      INTO CONTA
11          FROM UNITO U JOIN ASSOCIATO A1 ON  U.NOME_EXCHANGE_U = :NEW
        .NOME_EXCHANGE_DEX AND U.INDIRIZZO_U IN A1.INDIRIZZO_A
12              WHERE A1.ID_CONTO_VIRTUALE_A = :NEW.ID_CONTO_VIRTUALE_S
         AND A1.NOME_PIATTAFORMA_A = :NEW.NOME_PIATTAFORMA_S;
13
14      IF CONTA < 1 THEN
15          RAISE NON_COLLEGATO;
16      END IF;
17
18      EXCEPTION WHEN NON_COLLEGATO THEN
19          RAISE_APPLICATION_ERROR(-8000,'QUESTO SCAMBIO NON PUO
        ESSERE EFFETTUATO');
20
21  END CONTROLLO_SCAMBIO;
```

2.4.9 EFFETTUA_SCAMBIO

Prima di effettuare uno scambio viene controllato se sono disponibili i fondi affinchè questo accada. Se bisogna scambiare una valuta, verrà controllato il saldo tramite SALDOVAL, altrimenti se viene scambiata una cryptovaluta allora verrà richiamata la function SALDOCRYPTO.

```
CREATE OR REPLACE TRIGGER EFFETTUA_SCAMBIO
BEFORE INSERT ON SCAMBIO
FOR EACH ROW
DECLARE
    SALDO_INSUFFICIENTE EXCEPTION;
    SALDOSCAMBIOCRYPTO NUMBER := 0;
    SALDO NUMBER := 0;
    VALUTA_ VARCHAR(20);
BEGIN

    SELECT VALUTA
    INTO VALUTA_
    FROM CONTO_VIRTUALE
    WHERE ID_CONTO_VIRTUALE = :NEW.ID_CONTO_VIRTUALE_S AND
    NOME_PIATTAFORMA_CV = :NEW.NOME_PIATTAFORMA_S;

    IF :NEW.VALUTA_OTTENUTA = VALUTA_ OR :NEW.VALUTA_SCAMBIATA =
    VALUTA_ THEN
        SALDO := SALDOVAL(:NEW.ID_CONTO_VIRTUALE_S,:NEW.
    NOME_PIATTAFORMA_S,VALUTA_);
    ELSE
        SALDO := SALDOCRYPTO(:NEW.ID_CONTO_VIRTUALE_S,:NEW.
    NOME_PIATTAFORMA_S,:NEW.VALUTA_SCAMBIATA);
    END IF;

    DBMS_OUTPUT.PUT_LINE('SALDO: ' || SALDO);

    IF SALDO < :NEW.QUANTITA_SCAMBIATA THEN
        RAISE SALDO_INSUFFICIENTE;
    END IF;

    EXCEPTION WHEN SALDO_INSUFFICIENTE THEN
        RAISE_APPLICATION_ERROR(-9000,'SALDO INSUFFICIENTE!');

END EFFETTUA_SCAMBIO;
```

2.5 Procedure

Le procedure vengono utilizzate per automatizzare le operazioni che gli utenti possono compiere e per rendere la base di dati più ottimizzata possibile

2.5.1 ELIMINARE_CARTE_SCADUTE

Le carte scadute che non hanno versato almeno 500 euro(franchi o dollari) e che non sono collegate da almeno un anno vengono eliminate. Vengono selezionate innanzitutto tutte le carte scadute. Viene poi selezionato il conto virtuale associato a quella carta e vengono eseguiti i vari controlli. Utilizza per il totale dei depositi la function TOTDEPOSITI che prende in input la chiave primaria del conto virtuale. Dopo aver eliminato una carta di credito, viene controllato se sul conto virtuale ci siano collegate altre carte di credito, in caso contrario il conto virtuale viene eliminato.

```
CREATE OR REPLACE PROCEDURE ELIMINARE_CARTE_SCADUTE
```

```
IS
    DATACREAZIONE_  DATE;
    TOTDEPOSITI_ NUMBER;
    CONTATORE_ NUMBER;
    CONTROLLO NUMBER := 0;
BEGIN
    FOR i IN
        (
            SELECT NUMERO_CARTA_C,NOME_PIATTAFORMA_C,
    ID_CONTO_VIRTUALE_C
                FROM COLLEGATO C1 JOIN CARTA_DI_CREDITO CDC ON C1.
    NUMERO_CARTA_C = CDC.NUMERO_CARTA
                    WHERE CDC.DATA_DI_SCADENZA < SYSDATE
        )
    LOOP

        SELECT DATA_CREAZIONE
        INTO DATACREAZIONE_
        FROM CONTO_VIRTUALE
        WHERE ID_CONTO_VIRTUALE = i.ID_CONTO_VIRTUALE_C AND
    NOME_PIATTAFORMA_CV = i.NOME_PIATTAFORMA_C;

        TOTDEPOSITI_ := TOTDEPOSITO(i.ID_CONTO_VIRTUALE_C,i.
    NOME_PIATTAFORMA_C);

        IF TOTDEPOSITI_ < 500 AND (FLOOR((SYSDATE - DATACREAZIONE_)
     / 365.25) < 1) THEN
            DELETE FROM CARTA_DI_CREDITO WHERE NUMERO_CARTA = i.
    NUMERO_CARTA_C;
            DELETE FROM COLLEGATO WHERE NUMERO_CARTA_C = i.
    NUMERO_CARTA_C AND ID_CONTO_VIRTUALE_C = i.ID_CONTO_VIRTUALE_C
    AND NOME_PIATTAFORMA_C = i.NOME_PIATTAFORMA_C;
            DBMS_OUTPUT.PUT_LINE('E STATA CANCELLATA LA CARTA
    NUMERO: ' || i.NUMERO_CARTA_C);

            CONTROLLO := 1;

            SELECT COUNT(NUMERO_CARTA_C)
            INTO CONTATORE_
            FROM COLLEGATO
            WHERE NUMERO_CARTA_C = i.NUMERO_CARTA_C AND
    ID_CONTO_VIRTUALE_C = i.ID_CONTO_VIRTUALE_C AND
    NOME_PIATTAFORMA_C = i.NOME_PIATTAFORMA_C;

            IF CONTATORE_ < 1 THEN
                DELETE FROM CONTO_VIRTUALE WHERE ID_CONTO_VIRTUALE
    = i.ID_CONTO_VIRTUALE_C AND NOME_PIATTAFORMA_CV = i.
    NOME_PIATTAFORMA_C;
                DBMS_OUTPUT.PUT_LINE('E STATO CANCELLATO IL CONTO
    VIRTUALE CON ID: ' || i.ID_CONTO_VIRTUALE_C || ' E NOME: ' || i
    .NOME_PIATTAFORMA_C);
            END IF;

        END IF;

    END LOOP;

```

```
44      IF CONTROLLO = 0 THEN
45          DBMS_OUTPUT.PUT_LINE('NON E STATA CANCELLATA NESSUNA CARTA!
        ');
46      END IF;
47
48  END ELIMINARE_CARTE_SCADUTE;
```

2.5.2 PREMIO

Rilascia una reward all'utente che ha versato di più su ogni singolo conto. Il conteggio viene effettuato su ogni conto virtuale posseduti dall'utente tramite la functio TOTDEPOSITI. Si noti la query all'interno del loop in cui c'è un blocco che permette di ritornare una sola tupla poichè anche se esistono più carte di credito appartengono sempre allo stesso utente. La quantità di rewards rilasciate è l'1% dei depositi effettuati.

```
1   CREATE OR REPLACE PROCEDURE PREMIO
2   IS
3       TOTDEP_ NUMBER;
4       MASSIMO NUMBER := 0;
5       NOM_    VARCHAR(30);
6       COGN_    VARCHAR(30);
7       TOKEN VARCHAR(20);
8       CONTRATTO_ CHAR(42);
9       QUANTITAR NUMBER;
10  BEGIN
11
12      FOR i IN (
13          SELECT ID_CONTO_VIRTUALE,NOME_PIATTAFORMA_CV
14          FROM CONTO_VIRTUALE
15      )
16      LOOP
17
18          TOTDEP_ := TOTDEPOSITO(i.ID_CONTO_VIRTUALE,i.
        NOME_PIATTAFORMA_CV);
19
20          IF MASSIMO < TOTDEP_ THEN
21              MASSIMO := TOTDEP_;
22          END IF;
23
24      END LOOP;
25
26      FOR j IN (
27          SELECT ID_CONTO_VIRTUALE,NOME_PIATTAFORMA_CV
28          FROM CONTO_VIRTUALE
29      )
30      LOOP
31
32          TOTDEP_ := TOTDEPOSITO(j.ID_CONTO_VIRTUALE,j.
        NOME_PIATTAFORMA_CV);
33
34          SELECT NOME,COGNOME
35          INTO NOM_,COGN_
36              FROM UTENTE U JOIN (
37                  SELECT CF_UTENTE
```

```
38                      FROM CARTA_DI_CREDITO CDC JOIN COLLEGATO C1 ON
    CDC.NUMERO_CARTA = C1.NUMERO_CARTA_C
39                      WHERE C1.ID_CONTO_VIRTUALE_C = j.
    ID_CONTO_VIRTUALE AND C1.NOME_PIATTAFORMA_C = j.
    NOME_PIATTAFORMA_CV
40                      FETCH FIRST 1 ROWS ONLY
41          ) CDC ON U.CF = CDC.CF_UTENTE;
42
43          SELECT TOKEN_RIFERIMENTO
44          INTO TOKEN
45          FROM PIATTAFORMA_EXCHANGE
46          WHERE j.NOME_PIATTAFORMA_CV = NOME_PIATTAFORMA;
47
48          IF MASSIMO = TOTDEP_ THEN
49
50          QUANTITAR := FLOOR(MASSIMO/100);
51
52              INSERT INTO REWARDS(QUANTITA_R,DATA_RILASCIO,NOME_R,
    NOME_PIATTAFORMA_R,ID_CONTO_VIRTUALE_R) VALUES
53              (QUANTITAR,SYSDATE+1,TOKEN,j.NOME_PIATTAFORMA_CV,j.
    ID_CONTO_VIRTUALE);
54
55              DBMS_OUTPUT.PUT_LINE(NOM_ || ' ' || COGN_ || ' HA VINTO
     UN PREMIO!');
56
57          END IF;
58
59      END LOOP;
60  END PREMIO;
```

2.5.3 PREMIO_SCAMBI

Rilascia una reward agli utenti che negli ultimi 30 giorni hanno effettuato almeno 10 operazioni di scambio. Si noti la query all'interno del loop in cui c'è un blocco che permette di ritornare una sola tupla poichè anche se esistono più carte di credito, esse sono possedute sempre allo stesso utente.

```
1   CREATE OR REPLACE PROCEDURE PREMIO_SCAMBI
2   IS
3       NOME VARCHAR(20);
4       COGNOME VARCHAR(20);
5       CONTATORE NUMBER := 0;
6       TOKEN_ VARCHAR(20);
7   BEGIN
8
9       FOR i IN (
10          SELECT ID_CONTO_VIRTUALE,NOME_PIATTAFORMA_CV
11          FROM CONTO_VIRTUALE
12      )
13      LOOP
14
15          SELECT NOME,COGNOME
16          INTO NOME,COGNOME
17          FROM UTENTE
18          WHERE CF IN (
19              SELECT CF_UTENTE
```

```
            FROM CARTA_DI_CREDITO
            WHERE NUMERO_CARTA IN (
                SELECT NUMERO_CARTA_C
                FROM COLLEGATO
                WHERE ID_CONTO_VIRTUALE_C = i.ID_CONTO_VIRTUALE AND
 NOME_PIATTAFORMA_C = i.NOME_PIATTAFORMA_CV
                FETCH FIRST 1 ROWS ONLY
            )
        );

        SELECT TOKEN_RIFERIMENTO
        INTO TOKEN_
        FROM PIATTAFORMA_EXCHANGE
        WHERE NOME_PIATTAFORMA = i.NOME_PIATTAFORMA_CV;

        SELECT COUNT(TX_HASH)
        INTO CONTATORE
        FROM SCAMBIO
        WHERE ID_CONTO_VIRTUALE_S = i.ID_CONTO_VIRTUALE AND
    NOME_PIATTAFORMA_S = i.NOME_PIATTAFORMA_CV AND (SYSDATE -
    DATA_S) <= 30;

        IF CONTATORE >= 10 THEN
            INSERT INTO REWARDS(QUANTITA_R,DATA_RILASCIO,NOME_R,
    NOME_PIATTAFORMA_R,ID_CONTO_VIRTUALE_R) VALUES
            (CONTATORE,SYSDATE+1,TOKEN_,i.NOME_PIATTAFORMA_CV,i.
    ID_CONTO_VIRTUALE);

            DBMS_OUTPUT.PUT_LINE('UTENTE ' || NOME || ' ' ||
    COGNOME || ' HA EFFETTUATO '|| CONTATORE || ' SCAMBI, HA VINTO
    UN PREMIO CHE VERRA RILASCIATO DOMANI!');
        ELSE
            DBMS_OUTPUT.PUT_LINE('UTENTE ' || NOME || ' ' ||
    COGNOME || ' HA EFFETTUATO '|| CONTATORE || ' SCAMBI, NON HA
    VINTO NESSUN PREMIO!');
        END IF;

    END LOOP;

END PREMIO_SCAMBI;
```

2.5.4 AUMENTA_RISCHIO

Aumenta il rischio in VALUTAZIONE_RISCHIO se ha in almeno un conto virtuale un saldo maggiore o uguale a 500 euro(franchi o dollari).

```
CREATE OR REPLACE PROCEDURE AUMENTA_RISCHIO(CODICEFISCALE IN
    VARCHAR)
IS
    VALUTA VARCHAR(20);
    SALDO NUMBER;
    RIS NUMBER;
    AU_RIS NUMBER;
    CATE VARCHAR(20);
    CONTROLLO NUMBER := 0;
BEGIN
```

```
10
11      FOR i IN (
12          SELECT ID_CONTO_VIRTUALE_C,NOME_PIATTAFORMA_C
13              FROM COLLEGATO C1 JOIN CARTA_DI_CREDITO CDC ON C1.
    NUMERO_CARTA_C = CDC.NUMERO_CARTA
14                  WHERE CDC.CF_UTENTE = CODICEFISCALE
15      )
16      LOOP
17
18          SELECT TOKEN_RIFERIMENTO
19          INTO VALUTA
20          FROM PIATTAFORMA_EXCHANGE
21          WHERE NOME_PIATTAFORMA = i.NOME_PIATTAFORMA_C;
22
23          SALDO := SALDOVAL(i.ID_CONTO_VIRTUALE_C,i.
    NOME_PIATTAFORMA_C,VALUTA);
24
25          SELECT RISCHIO,AUMENTO_RISCHIO,CATEGORIA
26          INTO RIS,AU_RIS,CATE
27          FROM VALUTAZIONE_RISCHIO
28          WHERE VALUTAZIONE_RISCHIO.CF_UTENTE = CODICEFISCALE;
29
30          IF SALDO >= 500 THEN
31              CONTROLLO := 1;
32          END IF;
33
34      END LOOP;
35
36      IF CONTROLLO = 1 THEN
37          UPDATE VALUTAZIONE_RISCHIO SET RISCHIO = RIS + AU_RIS WHERE
     CATEGORIA = CATE AND VALUTAZIONE_RISCHIO.CF_UTENTE =
    CODICEFISCALE;
38          COMMIT;
39          DBMS_OUTPUT.PUT_LINE('IL TUO RISCHIO E AUMENTATO');
40      ELSE
41          DBMS_OUTPUT.PUT_LINE('IL TUO RISCHIO NON E AUMENTATO');
42      END IF;
43
44  END AUMENTA_RISCHIO;
```

2.5.5 ESTRATTO

Visualizza l'estratto conto dei movimenti e degli scambi compiuti dal trader. Essa non effettua nessuna modifica nelle tabelle, prende in input il codice fiscale dell'utente dato che per motivi di privacy i movimenti sono privati, a differenza degli scambi.

```
1   CREATE OR REPLACE PROCEDURE ESTRATTO(CODFISCALE IN VARCHAR)
2   IS
3       MOV VARCHAR(20);
4   BEGIN
5
6       DBMS_OUTPUT.PUT_LINE('ESTRATTO CONTO:');
7       FOR i IN (
8           SELECT TIPO_MOVIMENTO,QUANTITA_M,DATA_M,NUMERO_CARTA_M,
    FEE_M
```

```
9               FROM MOVIMENTO M1 JOIN CARTA_DI_CREDITO CDC ON M1.
    NUMERO_CARTA_M = CDC.NUMERO_CARTA
10                  WHERE CDC.CF_UTENTE = CODFISCALE
11                      ORDER BY DATA_M ASC
12      )
13      LOOP
14          IF i.TIPO_MOVIMENTO = 1 THEN
15              MOV := 'DEPOSITO';
16          ELSE
17              MOV := 'PRELIEVO';
18          END IF;
19
20          DBMS_OUTPUT.PUT_LINE(MOV || ':' || i.QUANTITA_M || '  ' ||
    ' DATA:' || i.DATA_M || ' ' || ' COMMISSIONI:' || i.FEE_M || '
    CARTA:' || i.NUMERO_CARTA_M);
21      END LOOP;
22
23      DBMS_OUTPUT.PUT_LINE('ESTRATTO CRYPTO:');
24      FOR j IN (
25          SELECT VALUTA_OTTENUTA,QUANTITA_OTTENUTA,VALUTA_SCAMBIATA,
    QUANTITA_SCAMBIATA,DATA_S,FEE_S
26              FROM SCAMBIO S JOIN (
27                  SELECT ID_CONTO_VIRTUALE_C
28                      FROM COLLEGATO C1 JOIN CARTA_DI_CREDITO CDC ON
    C1.NUMERO_CARTA_C = CDC.NUMERO_CARTA
29                          WHERE CDC.CF_UTENTE = CODFISCALE
30              ) C1 ON S.ID_CONTO_VIRTUALE_S = C1.ID_CONTO_VIRTUALE_C
31                  WHERE NOME_PIATTAFORMA_S IN (
32                      SELECT NOME_PIATTAFORMA_C
33                          FROM COLLEGATO C2 JOIN CARTA_DI_CREDITO CDC
     ON C2.NUMERO_CARTA_C = CDC.NUMERO_CARTA
34                              WHERE CDC.CF_UTENTE = CODFISCALE
35                  ) ORDER BY DATA_S ASC
36      )
37      LOOP
38          DBMS_OUTPUT.PUT_LINE(j.VALUTA_OTTENUTA || ':' || j.
    QUANTITA_OTTENUTA || ' ACQUISTATI CON: ' || j.VALUTA_SCAMBIATA
    || ':' || j.QUANTITA_SCAMBIATA || ' FEE PAGATE:' || j.FEE_S ||'
     IN DATA:' || j.DATA_S);
39      END LOOP;
40
41  END ESTRATTO;
```

2.6 Function

Una funzione è una serie di statement PL/SQL che, esattamente come le procedure, può essere richiamata tramite nome. La differenza tra le due però è che le funzioni riotrnano un valore all'ambiente in cui sono chiamate. In questo caso la chiamata alle function avviene nelle stored procedure, per rendere più pulito ed elegante il codice.

2.6.1 SALDOVAL

Funzione che calcola il saldo delle operazioni data la chiave del conto virtuale e la valuta impostata, tenendo conto di tutte le transazioni compiute sia nei movimenti che negli scambi.

```
CREATE OR REPLACE FUNCTION SALDOVAL(IDCONTOVIRTUALE IN VARCHAR,
    NOMEPIATTAFORMA IN VARCHAR,VALUTA_ IN VARCHAR)
RETURN NUMBER
IS
    SALDO NUMBER := 0;
    SALDO_VAL_MOV NUMBER := 0;
    SALDO_VAL_SCA NUMBER := 0;
BEGIN

    FOR i IN (
        SELECT *
        FROM MOVIMENTO
        WHERE ID_CONTO_VIRTUALE_M = IDCONTOVIRTUALE AND
    NOME_PIATTAFORMA_M = NOMEPIATTAFORMA
        ORDER BY DATA_M ASC
    )
    LOOP
        IF i.TIPO_MOVIMENTO = 1 THEN
            SALDO_VAL_MOV := SALDO_VAL_MOV + (i.QUANTITA_M - i.
    FEE_M);
        ELSE
            SALDO_VAL_MOV := SALDO_VAL_MOV - (i.QUANTITA_M - i.
    FEE_M);
        END IF;
    END LOOP;

    FOR j IN (
        SELECT *
        FROM SCAMBIO
        WHERE ID_CONTO_VIRTUALE_S = IDCONTOVIRTUALE AND
    NOME_PIATTAFORMA_S = NOMEPIATTAFORMA
        ORDER BY DATA_S ASC
    )
    LOOP
        IF j.VALUTA_OTTENUTA = VALUTA_ THEN
            SALDO_VAL_SCA := SALDO_VAL_SCA + (j.QUANTITA_OTTENUTA -
     j.FEE_S);
        ELSIF j.VALUTA_SCAMBIATA = VALUTA_ THEN
            SALDO_VAL_SCA := SALDO_VAL_SCA - (j.QUANTITA_SCAMBIATA
    - j.FEE_S);
        END IF;
    END LOOP;

    SALDO := SALDO_VAL_MOV + SALDO_VAL_SCA;

    RETURN SALDO;

END SALDOVAL;
```

2.6.2 SALDOCRYPTO

Calcola il saldo delle cryptovalute data la chiave del conto virtuale e la valuta impostata, tenendo conto degli scambi e delle reward ricevute.

```
CREATE OR REPLACE FUNCTION SALDOCRYPTO(IDCONTOVIRTUALE IN VARCHAR,
    NOMEPIATTAFORMA IN VARCHAR,VALUTA IN VARCHAR)
RETURN NUMBER
IS
    SALDO NUMBER := 0;
    SALDO_VAL_REW NUMBER := 0;
    SALDO_VAL_SCA NUMBER := 0;
BEGIN

    FOR k IN (
        SELECT *
        FROM SCAMBIO
        WHERE (VALUTA_SCAMBIATA = VALUTA OR VALUTA_OTTENUTA =
    VALUTA) AND ID_CONTO_VIRTUALE_S = IDCONTOVIRTUALE AND
    NOME_PIATTAFORMA_S = NOMEPIATTAFORMA
    )
    LOOP
        IF k.VALUTA_OTTENUTA = VALUTA THEN
            SALDO_VAL_SCA := SALDO_VAL_SCA + (k.QUANTITA_OTTENUTA -
     k.FEE_S);
        ELSE
            SALDO_VAL_SCA := SALDO_VAL_SCA - (k.QUANTITA_SCAMBIATA
    - k.FEE_S);
        END IF;
    END LOOP;

    FOR i IN (
        SELECT *
        FROM REWARDS
        WHERE NOME_R = VALUTA AND ID_CONTO_VIRTUALE_R =
    IDCONTOVIRTUALE AND NOME_PIATTAFORMA_R = NOMEPIATTAFORMA AND
    DATA_RILASCIO > SYSDATE+1
    )
    LOOP
        SALDO_VAL_REW := SALDO_VAL_REW + i.QUANTITA_R;
    END LOOP;

    SALDO := SALDO_VAL_SCA + SALDO_VAL_REW;

    RETURN SALDO;

END SALDOCRYPTO;
```

2.6.3 TOTDEPOSITO

Conteggia tutti i depositi data la chiave del conto virtuale.

```
CREATE OR REPLACE FUNCTION TOTDEPOSITO(IDCONTOVIRTUALE IN VARCHAR,
    NOMEPIATTAFORMA IN VARCHAR)
RETURN NUMBER
IS
    RETURNVALUE NUMBER;
```

```
BEGIN

    RETURNVALUE := 0;

    FOR i IN (
        SELECT QUANTITA_M,TIPO_MOVIMENTO
        FROM MOVIMENTO
        WHERE IDCONTOVIRTUALE = ID_CONTO_VIRTUALE_M AND
    NOMEPIATTAFORMA = NOME_PIATTAFORMA_M
    )
    LOOP
        IF i.TIPO_MOVIMENTO = 1 THEN
            RETURNVALUE := RETURNVALUE + i.QUANTITA_M;
        END IF;
    END LOOP;

    RETURN RETURNVALUE;

END TOTDEPOSITO;
```

2.7 Viste

Le Viste risultano molto utili per rendere i dati del database più fruibile dagli utenti. Nel nostro caso è stata crerata una sola vista che mostra tutti gli scambi effettuati da tutti gli utenti con i relativi Exchange Decentralizzati e la blockchain su cui avviene lo scambio. Questo è possibile poichè tutte le operazioni di scambio sono pubbliche dunque ogni utente può vedere gli scambi effettuati da se stesso e da tutti gli altri utenti del database. É stato deciso di non inserire altre viste per motivi di privacy, poichè ad esempio un utente non può visionare i prelievi e i depositi di altri utenti.

```
DROP VIEW ESTRATTO_CRYPTO;

CREATE OR REPLACE VIEW ESTRATTO_CRYPTO AS
    SELECT NOME_EXCHANGE,BLOCKCHAIN,VALUTA_OTTENUTA,
    QUANTITA_OTTENUTA,VALUTA_SCAMBIATA,QUANTITA_SCAMBIATA,DATA_S
        FROM EXCHANGE_DECENTRALIZZATO DEX JOIN SCAMBIO S ON DEX.
    NOME_EXCHANGE = S.NOME_EXCHANGE_DEX
            ORDER BY DATA_S ASC;
```

2.8 Data Control Language

Il Data Control Language riguarda la gestione degli utenti e dei loro permessi. Come stato già anticipato, in questo databese l'amministratore ha tutti i permessi. Il Gestore Conto invece gestisce tutto ciò che riguarda il conto virtuale mentre il Trader che è colui per cui è progettato il database puà effettuare la SELECT su tutto e in più può modificare i metodi di pagamento.

```
GRANT CONNECT, CREATE SESSION TO GESTORECONTO;
GRANT SELECT,UPDATE,DELETE ON CONTO_VIRTUALE TO GESTORECONTO;
GRANT SELECT,UPDATE,DELETE ON COLLEGATO TO GESTORECONTO;
```

```
4  GRANT SELECT,UPDATE,DELETE ON ASSOCIATO TO GESTORECONTO;
5  GRANT SELECT ON MOVIMENTO TO GESTORECONTO;
6  GRANT SELECT ON REWARDS TO GESTORECONTO;
7  GRANT SELECT ON SCAMBIO TO GESTORECONTO;
8  GRANT SELECT ON CRYPTO TO GESTORECONTO;
9  GRANT SELECT ON ESTRATTO_CRYPTO TO GESTORECONTO;--
10 GRANT EXECUTE ON ELIMINARE_CARTE_SCADUTE TO GESTORECONTO;
11 GRANT EXECUTE ON PREMIO TO GESTORECONTO;
12 GRANT EXECUTE ON PREMIO_SCAMBI TO GESTORECONTO;
13
14
15 GRANT CONNECT, CREATE SESSION TO TRADER;
16 GRANT SELECT,UPDATE ON VALUTAZIONE_RISCHIO TO TRADER;
17 GRANT SELECT ON UTENTE TO TRADER;
18 GRANT SELECT ON CRYPTO TO TRADER;
19 GRANT SELECT ON PIATTAFORMA_EXCHANGE TO TRADER;
20 GRANT SELECT ON CONTO_VIRTUALE TO TRADER;
21 GRANT SELECT ON COLLEGATO TO TRADER;
22 GRANT SELECT ON REWARDS TO TRADER;
23 GRANT SELECT ON WALLET_DECENTRALIZZATO TO TRADER;
24 GRANT SELECT ON BLOCKCHAIN TO TRADER;
25 GRANT SELECT ON ASSOCIATO TO TRADER;
26 GRANT SELECT ON EXCHANGE_DECENTRALIZZATO TO TRADER;
27 GRANT SELECT ON UNITO TO TRADER;
28 GRANT SELECT ON SCAMBIO TO TRADER;
29 GRANT SELECT ON ESTRATTO_CRYPTO TO TRADER;--
30 GRANT UPDATE ON CARTA_DI_CREDITO TO TRADER;
31 GRANT EXECUTE ON AUMENTA_RISCHIO TO TRADER;
32 GRANT EXECUTE ON ESTRATTO TO TRADER;
```

2.9 Future implementazioni

Il database ha numerose possibili implementazioni. Potrebbe essere ad esempio implementato uno **scheduler** che ogni anno elimina gli utenti più inattivi o la possibilità di effettuare **trading** anche con **NFT**(non-fungible token), il limite è rappresentato solo dalla fantasia.