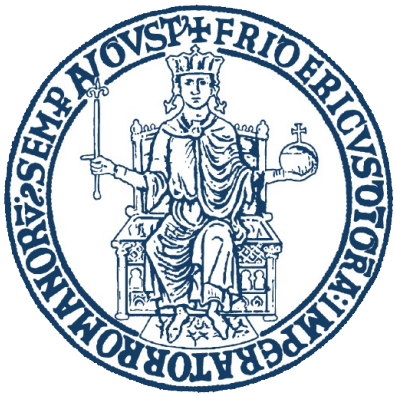
Progettazione e sviluppo di una Base Di Dati Relazionale per la gestione di SavingMoneyUnina

Davide Galdiero, Vincenzo Esposito

April 12, 2024



UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Indice

1 Progettazione concettuale **3**
- 1.1 Analisi dei requisiti . . . 3
- 1.2 Mappa Concettuale ER . . . 4
- 1.3 Mappa Concettuale UML . . . 5

2 Ristrutturazione della mappa concettuale **6**
- 2.1 Analisi delle ridondanze . . . 6
- 2.2 Eliminazione degli attributi multivalore . . . 6
- 2.3 Eliminazione degli attributi composti . . . 6
- 2.4 Analisi delle generalizzazioni . . . 6
- 2.5 Identificazione delle chiavi primarie . . . 6
- 2.6 Mappa Concettuale Ristrutturata UML . . . 7
- 2.7 Dizionario delle classi . . . 8
- 2.8 Dizionario delle Associazioni . . . 9

3 Traduzione al Modello Logico **9**
- 3.1 Mapping Associazioni . . . 9
  - 3.1.1 Associazioni 1-N . . . 9
- 3.2 Modello logico . . . 9

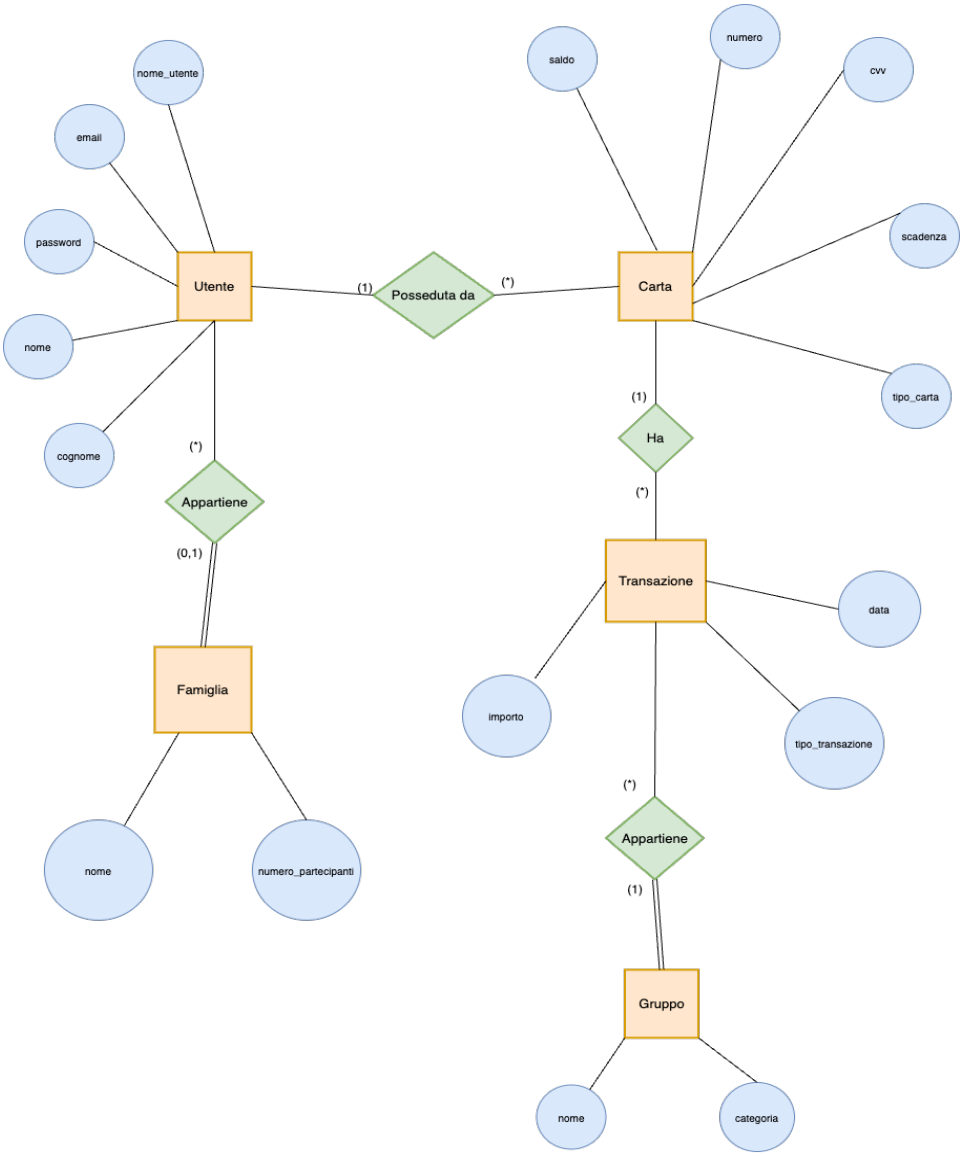
4 Progettazione Fisica **10**
- 4.1 Creazione Domini . . . 10
- 4.2 Creazione Tabelle . . . 10
- 4.3 Creazione Trigger . . . 12
- 4.4 Creazione Procedure . . . 13
- 4.5 Dizionario dei Vincoli . . . 13
  - 4.5.1 Vincoli di Dominio . . . 13
  - 4.5.2 Vincoli Intra-relazionali . . . 14
  - 4.5.3 Vincoli Inter-relazionali . . . 15

1 Progettazione concettuale

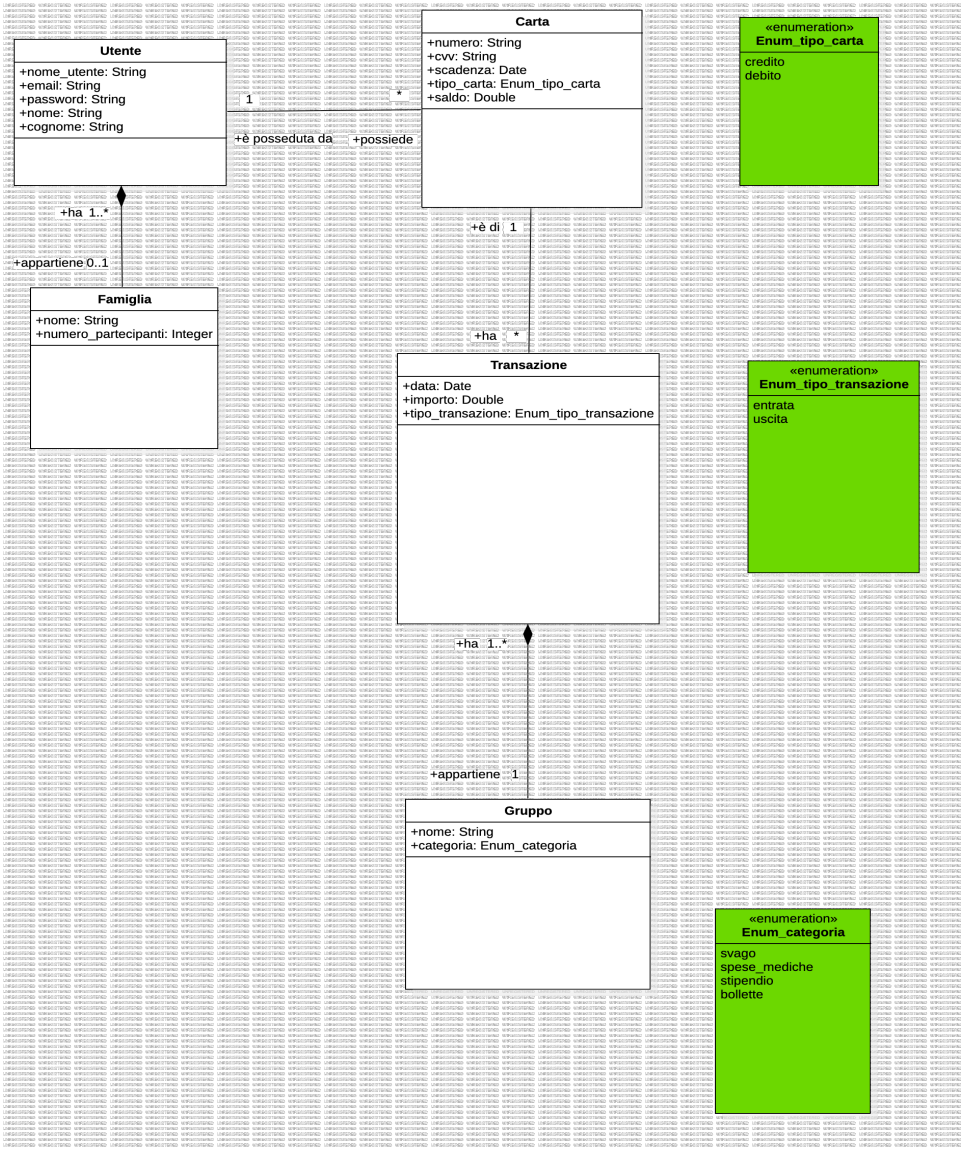
1.1 Analisi dei requisiti

La base di dati si deve occupare della gestione di un portafoglio. Deve permettere di collegare più carte di credito/debito.
Può essere condiviso in famiglia e divide le transazioni di ogni carta associata agli utenti in più categorie.
La figura centrale dello schema è la **Carta**.
Dell'entità Carta teniamo traccia del numero, della scadenza, del cvv e del tipo di carta per identificare la carta. In più teniamo traccia anche del saldo per una rapida gestione della carta.
Ogni Carta può avere delle **Transazioni** di cui è importante tenere traccia il tipo di transazione, data e importo. Ogni Transazione appartiene ad un **Gruppo** che si può formare se e solo se ha almeno una Transazione. Viene scelta questa implementazione per assegnare ad ogni transazione un Gruppo di cui teniamo traccia il nome e la categoria.
Ogni carta è posseduta da un **Utente** che è registrato nel Database e ognuno può appartenere ad una **Famiglia** per fare in modo di tenere traccia delle Carte di ogni Utente della Famiglia. Anche Famiglia ha senso se e solo se c'è almeno un Utente.

1.2 Mappa Concettuale ER



1.3 Mappa Concettuale UML



2 Ristrutturazione della mappa concettuale

2.1 Analisi delle ridondanze

C'è una ridondanza in Famiglia, ovvero, l'attributo *numero_partecipanti* da rimuovere per il nuovo modello concettuale, visto che possiamo calcolarlo sommando gli utenti appartenenti alla famiglia. Anche l'attributo *saldo* di Carta è una ridondanza dato che possiamo ricavarlo da somme e sottrazioni in Transazione, però nel nostro futuro applicativo accederemo spesso a saldo e quindi in questo caso è più conveniente lasciarlo.

2.2 Eliminazione degli attributi multivalore

Non sono presenti attributi multivalore.

2.3 Eliminazione degli attributi composti

Non sono presenti attributi composti.

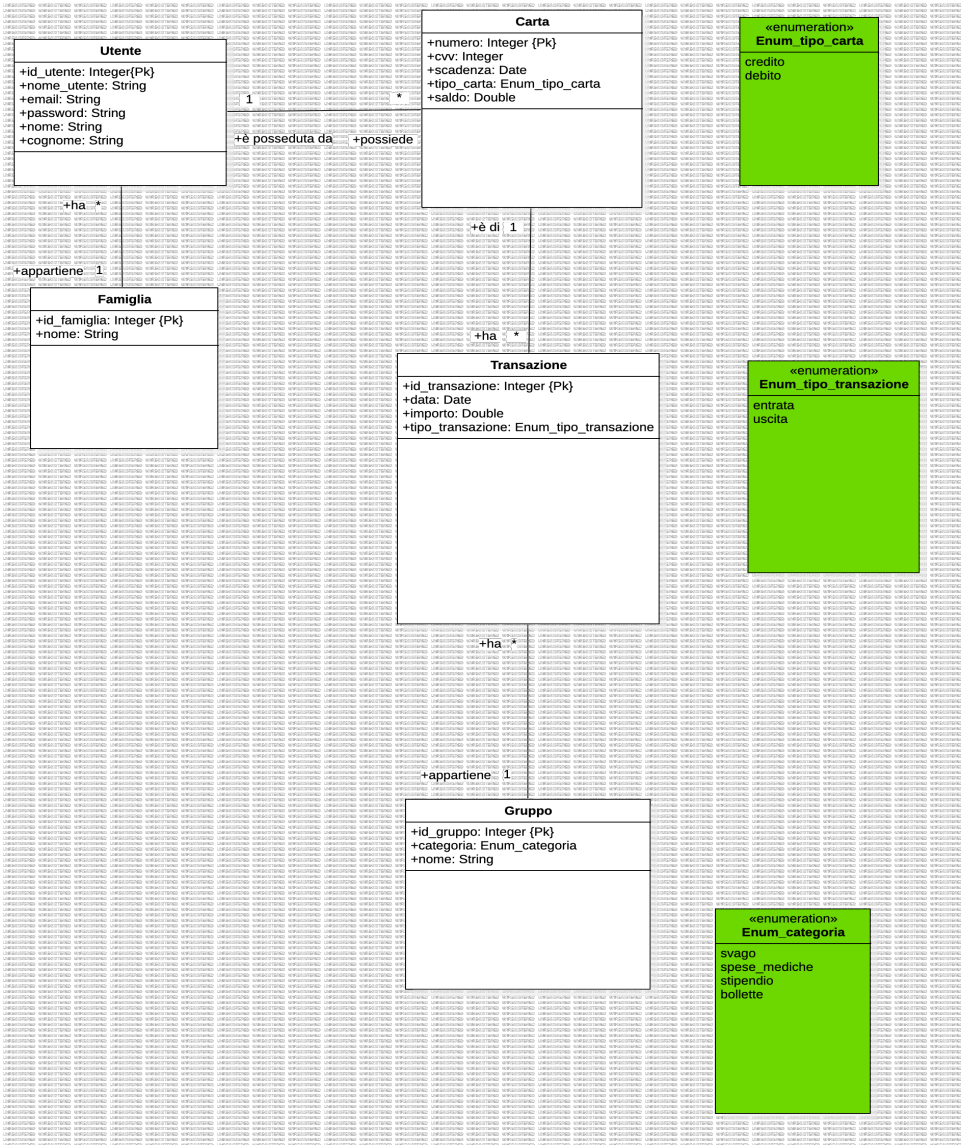
2.4 Analisi delle generalizzazioni

Famiglia e Gruppo sono in associazione di composizione rispettivamente con Utente e Transazione. A questo punto del modello non è necessaria mantenere queste generalizzazione, quello che è importante è la cardinalità delle associazioni. Quindi le rimuoviamo e le sostituiamo con associazioni *uno-a-molti*.

2.5 Identificazione delle chiavi primarie

- In Carta sarà l'attributo *numero*.
- In Utente viene aggiunto l'attributo *id_utente*.
- In Transazione viene aggiunto l'attributo *id_transazione*.
- In Gruppo viene aggiunto l'attributo *id_gruppo*.
- In Famiglia viene aggiunto l'attributo *id_famiglia*.

2.6 Mappa Concettuale Ristrutturata UML



2.7 Dizionario delle classi

Entità	Descrizione	Attributi
Carta	Carta generica	**numero** **scadenza** **cvv** **tipo_carta**: debito, credito **saldo**
Transazione	Entità che memorizza i movimenti della carta	**id_transazione** **data** **importo** **tipo_transazione**: entrata, uscita
Gruppo	Specializzazione di Transazione, ha senso se e solo se ha almeno una transazione	**id_gruppo** **categoria**: svago, spese mediche, stipendio, bollette **nome**
Utente	Generico utente iscritto al Database	**id_utente** **nome_utente** **email** **password** **nome** **cognome**
Famiglia	Specializzazione di utente, ha senso se e solo se almeno un utente appartiene a Famiglia	**id_famiglia** **nome**

2.8 Dizionario delle Associazioni

Associazioni	Descrizione	Attributi
ha	Associazione tra Carta e Transazione **molti-a-uno**	
posseduta da	Associazione tra Carta e Utente **uno-a-molti**	
appartiene	Associazione tra Transazione e Gruppo **uno-a-molti**	
appartiene	Associazione tra Utente e Famiglia **molti-a-uno**	

3 Traduzione al Modello Logico

3.1 Mapping Associazioni

3.1.1 Associazioni 1-N

- *Carta-ha-Transazione*: inserimento chiave di Carta in Transazione come chiave esterna.
- *Utente-possiede-Carta*: inserimento chiave di Utente in Carta come chiave esterna.
- *Gruppo-ha-Transazione*: inserimento chiave di Gruppo in Transazione come chiave esterna.
- *Famiglia-ha-Utente*: inserimento chiave di Famiglia in Utente come chiave esterna.

3.2 Modello logico

Gli attributi sottolineati sono chiavi primarie.
Gli attributi con un asterisco finale* sono chiavi esterne.

Carta (numero, cvv, scadenza, tipo_carta, saldo, id_utente*)

$id\_utente \rightarrow Utente.id\_utente$

Utente (id_utente, nome_utente, email, password, nome, cognome, id_famiglia*)

$id\_famiglia \rightarrow Famiglia.id\_famiglia$

Famiglia (id_famiglia, nome)

Transazione (id_transazione, data, importo, tipo_transazione, numero_carta*, id_gruppo*)

$numero\_carta \rightarrow Carta.numero$

$id\_gruppo \rightarrow Gruppo.id\_gruppo$

Gruppo (id_gruppo, categoria, nome)

4 Progettazione Fisica

4.1 Creazione Domini

```
    CREATE DOMAIN tipo_numero_carta AS CHAR(16)
CONSTRAINT check_numero_carta CHECK (LENGTH(VALUE) = 16);

    CREATE DOMAIN tipo_cvv AS VARCHAR(4)
CONSTRAINT check_cvv CHECK (LENGTH(VALUE) = 3 OR LENGTH(VALUE) = 4);

    CREATE DOMAIN Enum_tipo_carta AS VARCHAR(7)
CONSTRAINT check_tipo_carta CHECK (VALUE IN ('Debito', 'Credito'));

    CREATE DOMAIN Enum_tipo_transazione AS VARCHAR(7)
CONSTRAINT check_tipo_transazione CHECK (VALUE IN ('Entrata', 'Uscita'));

    CREATE DOMAIN Enum_categoria AS VARCHAR(13)
CONSTRAINT check_categoria CHECK (VALUE IN ('Svago', 'Spese_mediche',
'Stipendio', 'Bollette'));
```

4.2 Creazione Tabelle

Famiglia

```
    CREATE TABLE Famiglia(
        id_famiglia SERIAL,
        nome VARCHAR(20),

        CONSTRAINT pk_famiglia PRIMARY KEY (id_famiglia)
);
```

Utente

```
    CREATE TABLE Utente(
        id_utente SERIAL,
        nome_utente VARCHAR(20),
        email VARCHAR(50),
        password VARCHAR(20) NOT NULL,
        nome VARCHAR(20) NOT NULL,
```

```
        cognome VARCHAR(20) NOT NULL,
        id_famiglia INT,

        CONSTRAINT pk_utente PRIMARY KEY (id_utente),
        CONSTRAINT unique_nome_utente UNIQUE (nome_utente),
        CONSTRAINT unique_email UNIQUE (email),
        CONSTRAINT fk_id_famiglia FOREIGN KEY (id_famiglia)
        REFERENCES Famiglia (id_famiglia)
        ON UPDATE CASCADE
);
```

Carta

```
    CREATE TABLE Carta (
        numero tipo_numero_carta,
        cvv tipo_cvv NOT NULL,
        scadenza DATE NOT NULL,
        tipo_carta Enum_tipo_carta,
        saldo DECIMAL(10,2),
        id_utente INT NOT NULL,

        CONSTRAINT pk_numero PRIMARY KEY (numero),
        CONSTRAINT fk_id_utente FOREIGN KEY (id_utente)
        REFERENCES Utente (id_utente)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

Gruppo

```
    CREATE TABLE Gruppo(
        id_gruppo SERIAL,
        categoria Enum_categoria NOT NULL,
        nome VARCHAR(20),

        CONSTRAINT pk_gruppo PRIMARY KEY (id_gruppo)
);
```

Transazione

```
    CREATE TABLE Transazione(
        id_transazione SERIAL,
        data DATE NOT NULL DEFAULT CURRENT_DATE,
        importo DECIMAL(10,2) NOT NULL,
        tipo_transazione Enum_tipo_transazione,
        numero_carta tipo_numero_carta NOT NULL,
        id_gruppo INT,

        CONSTRAINT pk_id_transazione PRIMARY KEY (id_transazione),
        CONSTRAINT fk_numero_carta FOREIGN KEY (numero_carta)
        REFERENCES Carta (numero),
        CONSTRAINT fk_id_gruppo FOREIGN KEY (id_gruppo)
```

```
        REFERENCES Gruppo (id_gruppo)
        ON UPDATE CASCADE
);
```

4.3 Creazione Trigger

La descrizione dei vincoli è riportata nella sezione 4.5.3

aggiorna_saldo

```
    CREATE OR REPLACE FUNCTION aggiorna_saldo() RETURNS TRIGGER
LANGUAGE 'plpgsql'
AS $$
BEGIN
     IF (NEW.tipo_transazione = 'Entrata') THEN
          UPDATE Carta
               SET saldo = saldo + NEW.importo
               WHERE numero = NEW.numero_carta;
        ELSIF (NEW.tipo_transazione = 'Uscita') THEN
          UPDATE Carta
               SET saldo = saldo - NEW.importo
               WHERE numero = NEW.numero_carta;
        END IF;
        RETURN NEW;
END;
$$;

CREATE TRIGGER aggiorna_saldo
AFTER INSERT ON Transazione
FOR EACH ROW EXECUTE FUNCTION aggiorna_saldo();
```

verifica_saldo

```
    CREATE OR REPLACE FUNCTION verifica_saldo() RETURNS TRIGGER
LANGUAGE 'plpgsql'
AS $$
DECLARE
       current_saldo DECIMAL(10, 2);
BEGIN
     SELECT saldo INTO current_saldo
         FROM Carta
         WHERE numero = NEW.numero_carta;

         IF NEW.tipo_transazione = 'Uscita'
  AND NEW.importo > current_saldo THEN
               RAISE EXCEPTION 'Saldo insufficiente';
         END IF;
         RETURN NEW;
```

```
END;
$$;

CREATE TRIGGER verifica_saldo
BEFORE INSERT ON Transazione
FOR EACH ROW EXECUTE FUNCTION verifica_saldo();
```

4.4 Creazione Procedure

Per l'inserimento di una Famiglia viene progettata una Procedura memorizzata, cosicchè quando viene inserita si aggiorna anche il campo associato nella riga Utente interessata.

```
    CREATE OR REPLACE PROCEDURE
    inserisci_famiglia_aggiorna_utente(nome_famiglia VARCHAR(20),
    n_utente VARCHAR(20))
LANGUAGE 'plpgsql'
AS $$
DECLARE
       current_id_famiglia INT;
BEGIN
       INSERT INTO Famiglia(nome) VALUES(nome_famiglia)
           RETURNING id_famiglia INTO current_id_famiglia;

           UPDATE Utente SET id_famiglia = current_id_famiglia
           WHERE nome_utente = n_utente;
END;
$$;
```

4.5 Dizionario dei Vincoli

4.5.1 Vincoli di Dominio

Carta	
Vincolo	**Descrizione**
tipo_numero_carta	Il valore inserito deve avere lunghezza 16.
tipo_cvv	Il valore inserito deve avere lunghezza 3 o 4.
Enum_tipo_carta	Il valore inserito deve essere o Debito o Credito.

Transazione	
Vincolo	**Descrizione**
Enum_tipo_transazione	Il valore inserito deve essere Entrata o Uscita.

Categoria	
Vincolo	**Descrizione**
Enum_categoria	Il valore inserito deve essere Svago o Spese_mediche o Stipendio o Bollette.

4.5.2 Vincoli Intra-relazionali

Famiglia	
Vincolo	**Descrizione**
pk_famiglia	Il vincolo di chiave primaria.

Utente	
Vincolo	**Descrizione**
pk_utente	Il vincolo di chiave primaria.
unique_nome_utente	Il vincolo di unicità.
unique_email	Il vincolo di unicità.
fk_id_famiglia	Il vincolo di chiave esterna.

Carta	
Vincolo	**Descrizione**
pk_numero	Il vincolo di chiave primaria.
fk_id_utente	Il vincolo di chiave esterna.

Gruppo	
Vincolo	**Descrizione**
pk_gruppo	Il vincolo di chiave primaria.

Transazione	
Vincolo	**Descrizione**
pk_id_transazione	Il vincolo di chiave primaria.
fk_numero_carta	Il vincolo di chiave esterna.
fk_id_gruppo	Il vincolo di chiave esterna.

4.5.3 Vincoli Inter-relazionali

L'implementazione SQL dei seguenti trigger è riportata nella Sezione 4.3

- **aggiorna_saldo**
  Dopo ogni transazione aggiorna il saldo della carta in automatico.
- **verifica_saldo**
  Prima di ogni transazione verifica se c'è saldo sufficiente.