Relazione progetto Basi di Dati

Shop&fight

Viviana Alessio matr. 1029720
Giacomo Beltrame matr. 1006153

17 luglio 2014

Indirizzo web del sito

http://basidati.studenti.math.unipd.it/basidati/~gbeltram/

Credenziali d'accesso

Utente amministratore admin, admin

Utente senza privilegi utente, utente

Indirizzo e-mail referente

giacomo.beltrame@studenti.unipd.it

Indice

1 abstract

I proprietari di molti negozi fisici sentono il bisogno di avere un sito web che funga da vetrina virtuale e permetta di acquistare anche online i prodotti che si possono trovare nel vero negozio. Per questo abbiamo deciso di creare una base di dati che modella gli elementi necessari alla gestione di un sito di questo tipo. Quindi riuscirà a far interagire gli utenti con il databare attraverso un'intefaccia web che, ad esempio, permetterà loro di visualizzare e acquistare i prodotti presenti nel sito. Il nostro progetto si chiama Shop&fight e tratta prodotti per sport da combattimento.

2 analisi dei requisiti

Vogliamo creare un database che tratti la vendita di articoli sportivi, di seguito esponiamo cosa ci interessa di ogni entità.

Di ogni prodotto interessa il codice l'identificatico, il nome, il tipo, l'eventuale sottotipo, la marca, lo sport, la descrizione. vi sono poi tre tipoligie di prodotti: gli indumenti dei quali interessa la taglia e il sesso, l'attrezzatura della quale interessa peso, materiale, misure(HxLxW), livello(atleta a livello principiante, intermedio, avanzato), gli accessori dei quali interessano misure, peso, materiale.

Gli utenti sono divisi in due categorie: i clienti e i membri dello staff del negozio. Dei clienti interessa il codice identificativo, l'username il nome, il cognome, la password, l'indirizzo. Dei membri dello staff interessa il codice identificativo, il Nome, il Cognome.

Ogni utente ha una propria login che consiste dell' identificativo dell'utente e della sua password.

Gli ordini dei clienti sono divisi in confermati e non confermati e interessa di entrambi il codice identificativo, il codice del cliente, i codici dei prodotti. Degli ordini confermati interessa anche la data di conferma.

Gli utenti possono accumolare degli sconti cumulabili (ogni 50 euro 10 euro di sconto) che possono decidere quando utilizzare.

2.1 operazioni tipiche

I clienti possono visualizzare le caratteristiche dei prodotti e comprare prodotti. Vedere cos' hanno nel proprio carrello e cos'hanno in ordine. Possono inoltre accumolare sconti grazie ai loro acquisti.

I membri dello staff(Admin) vedono tutto il carrello e tutti gli ordini confermati. Possono inoltre eliminare gli utenti ed eliminare gli sconti.

3 Progettazione concettuale

3.1 Descrizione delle classi

Utenti

- IDu: int
- Nome: string
- Cognome: string
- Password: string

sottoclassi di Utenti: Admin, Clienti

Clienti

- Username: string
- Indirizzo: string

Admin

- Ruolo: string

Prodotti

- IDp: int
- NomeProdotto: string
- Tipo: string
- Sottotipo: string
- Marca: string
- Descrizione:string

- Prezzo: double

sottoclassi di Prodotti: Indumenti, Attrezzatura, Accessori.

Indumenti

- TagliaV: enum
- TagliaS: int
- Sesso: enum (uomo, donna, unisex)

Attrezzatura

- Peso: string
- Materiale: string
- Misure: string
- Livello: enum (principiante, intermedio, avanzato)

Accessori

- Peso: string
- Materiale: string
- Misure: string

Ordini

- IdOrdine
- IdUtente

sottoclassi di Ordini: Confermati, NonConfermati(la quale non ha nessun attributo significativo).

Confermati

- DataConferma: date

Sport

- NomeSport: string

Marche

- NomeMarca: string

3.2 Relazioni tra le classi

Sconti-Utenti contiene lo sconto accumolato di
- molteplicita: 1:1
- totalità: totale verso Utenti, parziale verso Sconti
- ogni sconto appartiene a un utente, ma non tutti gli utenti hanno sconti (gli admin o chi non ha ancora speso 100 euro)

Utenti-Ordini contiene gli ordini di
- molteciplità: 1:M
- totalità: totale verso Utenti, parziale verso Ordini
- ogni utente può avere molti ordini oppure nessuno, un ordine appartiene ad un solo utente

Ordini-Prodotti contiene
- molteplicità: N:M
- totalità: totale verso Prodotti, parziale verso Ordini
- un ordine può contenere più prodotti, un prodotto può essere contenuto in più ordini oppure

in nessuno

Prodotti-Sport è usato nello sport
- molteplicità: N:M
- totalità: totale in entrambi i versi
- un prodotto può essere utilizzato in diversi sport, ad uno sport possono essere associati uno o più prodotti

Prodotti-Marche appartiene
- molteplicità: M:1
- totalità: totale in entrambi i versi
- un prodotto appartiene a una sola marca, una marca ha uno o più prodotti

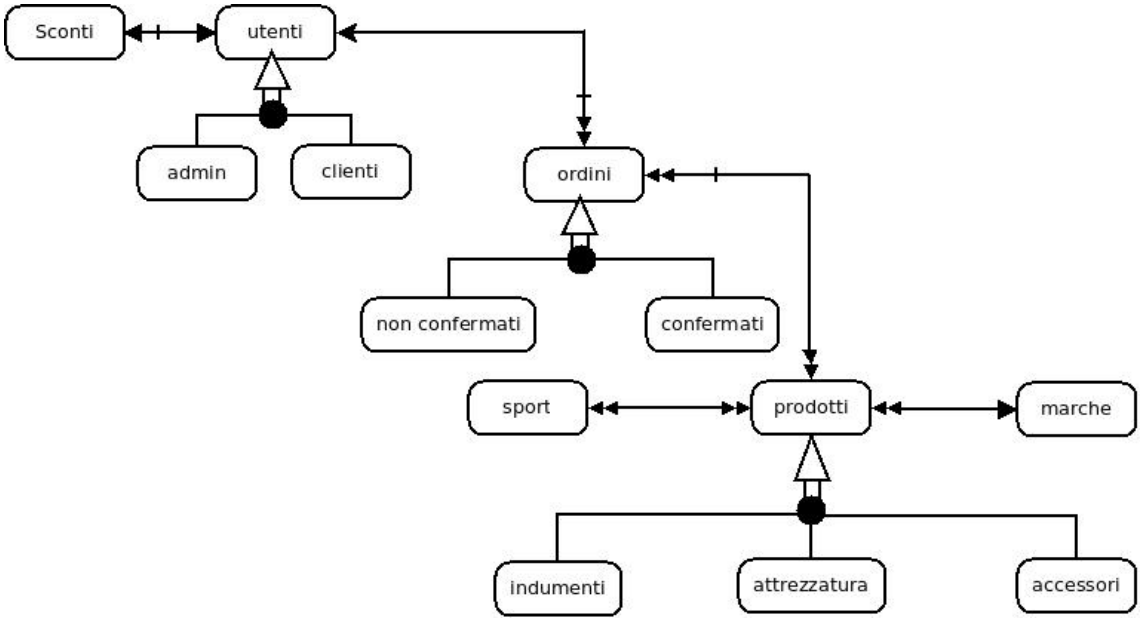

Figura 1: schema a oggetti

4 Progettazione logica

4.1 Creazione tabelle per relazioni N:M

Avendo nello schema a oggetti due relazioni molti a molti abbiamo dovuto creare nello schema logico due tabelle aggiuntive. Le relazioni molti a molti sono *ordini-prodotti* e *sport-prodotti* e abbiamo realizzato le tabelle Prodotti_Ordine e Prodotti_sport.

4.2 creazione tabella Prodotti_venduti

Abbiamo deciso di creare la tabella Prodotti_venduti perchè altrimenti avremmo avuto problemi quando un utente decide di comprare un articolo. Il desiderio di acquistare un prodotto comporta l'eliminazione del prodotto dalla tabella Prodotti, quindi successivamente non avremmo più avuto informazioni su quell'articolo. La tabella Prodotti_venduti contiene quindi tutti gli attributi della tabella Prodotti in modo tale da avere salvati i dati dei prodotti venduti. La cancellazione di un prodotto dalla tabella Prodotti è necessario in quanto altrimenti il prodotto risulterebbe ancora disponibile quando in realtà non lo è essendo già negli ordini di un utente.

Ci teniamo a precisare che questa tabella non contiene esattamente tutti gli attributi della tabella prodotti, ma solo quelli che abbiamo ritenuto necessari per poter capire esattamente quale prodotto è stato venduto.

4.3 Creazione tabella Carrello

Durante la progettazione logica abbiamo creato questa tabella per fornire all'utente uno step intermedio tra la scelta dei prodotti da comprare e l'acquisto vero e prorpio, proprio come avviene in molti siti di e-commerce. Nella tabella Carrello in sostanza sono presenti tutti i carrelli dei nostri utenti visto che si tratta di una tabella di passaggio un utente potrà avere solamente un prorpio carrello e questo sarà identificato dal suo IDu(id dell'utente). Gli attrinuti della tabella sono infatti IDu ed IDp, con IDp chiave primaria.

4.4 trasformazione gerarchie

Nello schema relazionale sono presenti tre gerarchie: utenti, ordini, prodotti. Abbiamo deciso di trasformarle come segue.

1. la gerarchia degli utenti diventa una tabella unica
2. la gerarchia degli ordini diventa una tabella unica
3. la gerarchia dei prodotti diventa un partizionamento verticale però gli accessori diventano parte dell'attrezzatura visto che secondo noi non ha attributi significativi per essere una tabella a se stante.

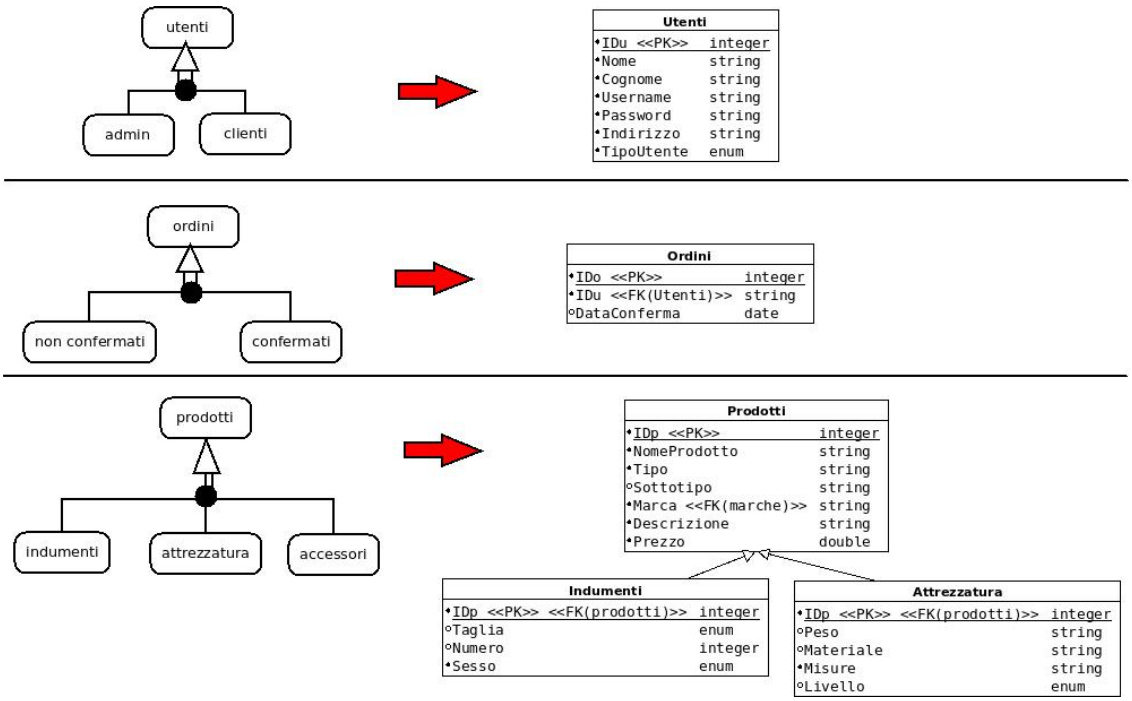

Figura 2: trasformazioni gerarchie

4.5 Descrizione completa delle tabelle

Utenti

- IDu: int PK
- Nome: string
- Cognome: string
- Username: string
- Password: string
- Indirizzo: string
- Tipoutente: (Cliente,Proprietario,Commesso)

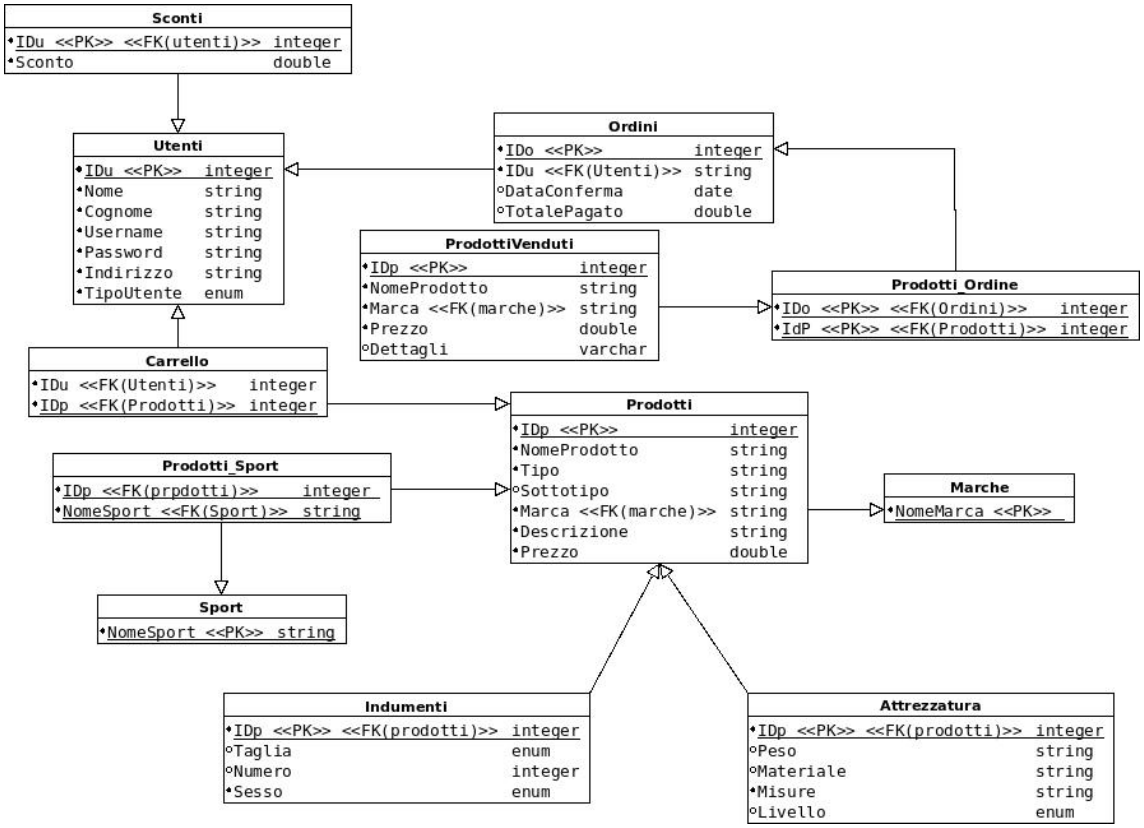

Figura 3: schema logico

Prodotti

- IDp: int PK
- NomeProdotto: string
- Tipo: string
- Sottotipo: string
- Marca: string FK(marche)
- Descrizione:stringSpor
- Prezzo: double

Indumenti

- IDp: int PK FK(prodotti)
- TagliaV: enum
- TagliaS: int
- Sesso: enum (uomo, donna, unisex)

Attrezzatura

- IDp: int PK FK(prodotti)
- Peso: string
- Materiale: string
- Misure: string
- Livello: enum (principiante, intermedio, avanzato)

Prodotti_venduti

- IDp: int PK
- NomeProdotto: string
- Tipo: string
- Sottotipo: string
- Marca: string
- Descrizione:string
- Prezzo: double

Carrello

- IDu FK(Utenti)
- IDp PK FK(Prodotti)

Ordini

- IdOrdine
- IdUtente
- DataConferma: date
- TotalePagato: double

Prodotti_Ordine

- IdOrdine: int PK FK(Ordini)
- IdP: int FK(Prodotto_venduti)

Prodotti_Sport

- IdP: int PK FK()
- NomeSport

Sport

- NomeSport: string

Marche

- NomeMarca: string

4.6 Chiavi esterne

sconti-utenti Alla tabella degli sconti serve sapere a quale utente associare un certo sconto quindi ha bisogno di una chiave esterna con l'IDu
ordini-utenti Alla tabella degli ordini serve sapere quale utente associare un certo ordine quindi ha bisogno di una chiave esterna con l'IDu
prodotti_ordine-ordini La tabella prodotti_ordine ha bisogno di sapere a quale ordine appartiene un determinato prodotto, quindi dell'IDo.
prodotti_ordine-prodottiVenduti La tabella prodottiVenduti ha bisogno di sapere quali prodotti presenti nella tabella dei prodotti degli ordini inserire nella tabella dei prodotti venduti, ha bisogno quindi dell'IDp.
Prodotti-Marche La tabella dei prodotti ricava dalla tabella delle marche la marca di un determinato prodotto, ha bisogno quindi del nome della marca.
Indumenti-Prodotti, Attrezzatura-Prodotti Entrambe le tabelle (Indumenti e Attrezzatura) essendo sottoclassi hanno bisogno dell'identificativo di un determinato prodotto (IDp).
prodotti_sport-Prodotti La tabella prodotti_sport ha bisogno di sapere a quale prodotto abbinare uno sport, ha bisogno quindi di IDp.

prodotti_sport-Sport La tabella prodotti_sport ha bisogno di sapere quali sport abbinare a quali prodotti, ha bisogno quindi di NomeSport.
Carrello-Prodotti Serve una chiave esterna per sapere i prodotti da abbinare all'utente giusto (IDp).
Carrello-Utenti Serve una chiave esterna per sapere l'utente "proprietario" del carrello (IDu).

5 Query, procedure, funzioni e trigger

5.1 Query significative

5.1.1 Query 1

Query che ritorna lo sport maggiormente presente tra i prodotti attualmente nel carrello.

```
DROP VIEW IF EXISTS ContaSport;

CREATE VIEW ContaSport AS
SELECT NomeSport, count(*) AS Num
FROM Carrello NATURAL JOIN Prodotti NATURAL JOIN Prodotti_sport
GROUP BY NomeSport;

SELECT NomeSport, MAX(Num)
FROM ContaSport;
```

NomeSport	MAX(Num)
BJJ	9

Figura 4: Risultato query 1

5.1.2 Query 2

Query che ritorna l'id dei clienti che hanno comprato solo prodotti di una marca o di un tipo in tutti gli ordini.

```
DROP VIEW IF EXISTS ContaMarca;

CREATE VIEW ContaMarca AS
SELECT DISTINCT idu, marca
FROM Utenti NATURAL JOIN Ordini NATURAL JOIN Prodotti_ordine
NATURAL JOIN Prodotti_venduti;

DROP VIEW IF EXISTS ContaTipo;

CREATE VIEW ContaTipo AS
SELECT DISTINCT idu, tipo
FROM Utenti NATURAL JOIN Ordini NATURAL JOIN Prodotti_ordine
NATURAL JOIN Prodotti_venduti;

SELECT idu
FROM ContaMarca
GROUP BY idu
HAVING count(*)=1
```

```
UNION

SELECT idu
FROM ContaTipo
GROUP BY idu
HAVING count(*)=1;
```

idu
4

Figura 5: Risultato query 2

5.1.3 Query 3

Query che ritorna l'username degli utenti che non hanno comprato niente.

```
SELECT IDu, Username
FROM Utenti
WHERE IDu NOT IN( SELECT IDu
                  FROM Ordini);
```

IDu	Username
5	andry45

Figura 6: Risultato query 3

5.1.4 Query 4

Query che ritorna l'utente che ha applicato lo sconto maggiore.

```
DROP VIEW IF EXISTS MostraDifferenza;

CREATE VIEW MostraDifferenza AS
SELECT idu, ido, SUM(Prezzo) AS TotaleOrdine, TotalePagato
FROM Utenti NATURAL JOIN Ordini NATURAL JOIN Prodotti_ordine
NATURAL JOIN Prodotti_venduti
GROUP BY ido;

SELECT idu
FROM MostraDifferenza
WHERE TotaleOrdine-TotalePagato = (SELECT MAX(TotaleOrdine-TotalePagato)
                                   FROM MostraDifferenza);
```

idu
2

Figura 7: Risultato query 4

5.1.5 Query 5

Query che ritorna utenti che hanno effettuato almeno un ordine il cui valore (non scontato, somma dei prezzi dei prodotti) è maggiore di 86 euro (usa la view *MostraDifferenza* del punto precedente).

```
SELECT DISTINCT idu
FROM MostraDifferenza
WHERE TotaleOrdine > 86;
```

idu
1
3
4

Figura 8: Risultato query 5

5.1.6 Query 6

Query che ritorna idutente, nome, cognome e il numero di sconti dell' utente ha applicato il maggior numero di sconti.

```
DROP VIEW IF EXISTS ContaSconti;

CREATE VIEW ContaSconti AS
SELECT DISTINCT idu, ido, TotalePagato
FROM Utenti NATURAL JOIN Ordini NATURAL JOIN Prodotti_ordine
NATURAL JOIN Prodotti_venduti
GROUP BY ido
HAVING SUM(Prezzo) <> TotalePagato;

SELECT idu, nome, cognome, count(*) AS NumeroSconti
FROM ContaSconti NATURAL JOIN Utenti
GROUP BY idu
HAVING NumeroSconti >= ALL (SELECT count(*)
                            FROM ContaSconti
                            GROUP BY idu);
```

idu	nome	cognome	NumeroSconti
1	daniele	rostirolla	3

Figura 9: Risultato query 6

5.1.7 Query 7

Query che ritorna username degli utenti che hanno ordinato almeno 3 prodotti in un ordine.

```
DROP VIEW IF EXISTS ContaProdotti;

CREATE VIEW ContaProdotti AS
SELECT ido, count(*) AS Prodotti
FROM Prodotti_ordine
```

```
GROUP BY IDo;

SELECT DISTINCT username
FROM Utenti NATURAL JOIN Ordini NATURAL JOIN ContaProdotti
WHERE Prodotti >3;
```

username
admin

Figura 10: Risultato query 7

5.1.8 Query 8

Query che ritorna gli utenti che hanno prodotti in ordine, ma non nel carrello.

```
SELECT Username
FROM Utenti
WHERE IDu NOT IN ( SELECT IDu
                   FROM Utenti NATURAL JOIN Carrello )
AND IDu IN ( SELECT IDu
             FROM Utenti NATURAL JOIN Ordini );
```

Username
albertino97
zanny89

Figura 11: Risultato query 8

5.2 Procedure

5.2.1 AddCart

Procedura che aggiunge un prodotto nel carrello dell'utente.

```
CREATE PROCEDURE AddCart (user VARCHAR(15),Idprod INT)
BEGIN

  DECLARE Idut INT;
  SELECT idu INTO Idut FROM Utenti WHERE username=user;
  INSERT INTO Carrello VALUES (Idut,Idprod);

END $
```

5.2.2 ConcludiAcquisto

Procedura che crea un nuovo ordine, inserisce i Prodotti da acquistare nella tabella Prodotti_venduti collegandoli opportunamente con l'ordine corrente.

```
CREATE PROCEDURE ConcludiAcquisto (IN idordine INT, idprodotto INT)
BEGIN

  DECLARE nomeprod VARCHAR(40);
  DECLARE tipoprod CHAR(20);
  DECLARE sottotipoprod CHAR(20);
  DECLARE marcaprod CHAR(20);
  DECLARE descrizioneprod VARCHAR(100);
  DECLARE prezzoprod DOUBLE(7,2);

  INSERT INTO Prodotti_ordine
  VALUES (idordine, idprodotto);

  SELECT nomeprodotto, tipo, sottotipo, marca, prezzo
  INTO nomeprod, tipoprod, sottotipoprod, marcaprod, prezzoprod
  FROM Prodotti
  WHERE idp=idprodotto;

  IF (tipoprod="Attrezzatura") THEN
    SELECT concat_ws(', ', peso, materiale, misure, livello)
    INTO descrizioneprod
    FROM Attrezzatura
    WHERE idp=idprodotto;
  ELSE
    SELECT concat_ws(', ', taglia, numero, sesso)
    INTO descrizioneprod
    FROM Indumenti
    WHERE idp=idprodotto;
  END IF;

  INSERT INTO Prodotti_venduti
  VALUES (idprodotto, nomeprod, tipoprod, sottotipoprod,
  marcaprod, descrizioneprod, prezzoprod);

END $
```

5.3 Funzioni

5.3.1 TotaleOrdine

Funzione che applica lo sconto all'ordine se l'utente lo richiede e setta a 0 lo sconto di conseguenza.

```
CREATE FUNCTION TotaleOrdine (idutente INT, applicasconto BOOLEAN)
RETURNS DOUBLE(5,2)
BEGIN

  DECLARE totale DOUBLE(5,2);
  DECLARE sconto INT;

  SELECT SUM(prezzo) INTO totale
  FROM Carrello natural join Prodotti
  WHERE idu=idutente;

  IF(applicasconto) THEN
    SELECT scontoaccumulato INTO sconto
    FROM Sconti WHERE idu=idutente;
```

```
    IF (totale > sconto) THEN
      SET totale = (totale - sconto);
    ELSE SET totale = 0;
    END IF;
    UPDATE Sconti SET scontoaccumulato=0 WHERE idu=idutente;
  END IF;
  RETURN totale;

END $
```

5.4 Trigger

5.4.1 NewDiscount

Trigger che dopo l'inserimento di un nuovo cliente ne crea il relativo record nella tabella Sconti. L'inserimento degli utenti non è previsto lato applicazione quindi il trigger si attiva soltanto al popolamento iniziale del database. Lo sconto inizialmente per default vale 0.

```
CREATE TRIGGER NewDiscount
AFTER INSERT ON Utenti
FOR EACH ROW
BEGIN

  INSERT INTO Sconti VALUES (NEW.idu,0);

END $
```

5.4.2 Discount

Trigger che si occupa di aggiornare il buono sconto di un cliente in base alla spesa effettuata (lo sconto è cumulabile).

```
CREATE TRIGGER Discount
AFTER INSERT ON Ordini
FOR EACH ROW
BEGIN

  DECLARE Idut INT;
  DECLARE Totale DOUBLE(5,2);

  SELECT DISTINCT NEW.idu INTO Idut
  FROM Ordini;

  SELECT DISTINCT NEW.totalepagato INTO Totale
  FROM Ordini;

  IF (Totale >= 50.00 AND Totale < 100.00) THEN
    UPDATE Sconti
    SET scontoaccumulato = (scontoaccumulato + 5)
    WHERE idu = Idut;
    ELSEIF (Totale >= 100.00 AND Totale < 150.00) THEN
      UPDATE Sconti
      SET scontoaccumulato = (scontoaccumulato + 10)
      WHERE idu = Idut;
      ELSEIF (Totale >= 150.00) THEN
        UPDATE Sconti
```

```
        SET scontoaccumulato = (scontoaccumulato + 15)
        WHERE idu = Idut;
    END IF;

END $
```

5.4.3 EliminaProdotto

Trigger che si occupa di eliminare il prodotto acquistato dopo l'inserimento dello stesso nella tabella Prodotti_venduti (le righe delle altre tabelle che corrispondono al prodotto interessato vengono eliminate automaticamente per la proprietà *ON DELETE CASCADE* delle chievi esterne).

```
CREATE TRIGGER EliminaProdotto
AFTER INSERT ON Prodotti_venduti
FOR EACH ROW
BEGIN

    DELETE FROM Prodotti
    WHERE idp=NEW.idp;

END $
```

6 Interfaccia web

L'interfaccia web con cui gli utenti possono interagire con la base di dati è strutturata in modo piuttosto semplice ed intuitivo in modo da facilitarne l'uso.
Costituita dal logo in alto a sinistra affiancato dal menu orizzontale delle pagine e sotto di esso l'indicazione di dove ci si trova nella navigazione del sito che essendo molto semplice solo poche volte scende di livello.
Tutte le pagine mostrano una form di login e registrazione nel caso non ci si sia ancora autenticati; questo permette di visualizzare il contenuto dell'applicazione solamente agli utenti registrati al sito.
Prima di passare in dettaglio alla trattazione delle pagine è doverosa una breve spiegazione delle caratteristiche dell'utenza dell'applicazione in questione.

Utenti: Innanzitutto precisiamo che vi sono due tipi di utenti che hanno privilegi differenti: i Clienti e gli Amministratori(che si suddividono in proprietari e commessi). I clienti, una volta che si sono registrati potranno accedere al sito attraverso la form di login. Ciò che distingue un Cliente da un Amministratore sta nel fatto che un cliente può vedere soltanto i propri ordini effettuati e lo sconto al momento a disposizione mentre l'amministratore(proprietario) ha la possibilità di vedere tutti gli ordini e gli sconti accumulati da ogni cliente, visualizzare e gestire i clienti(eliminare un profilo e azzerarne lo sconto a disposizione).

Descrizione delle pagine:

index.php: questa è la home page dove si può visualizzare l'username con cui si è registrati al sito, una breve descrizione di esso ed un link in basso per effettuare il logout.

filtra.php: genera la pagina *Cerca/Filtra* che permette di cercare il prodotto interessato usando i vari strumenti a disposizione. La form della pagina tramite una post reindirizza alla pagina stessa invia i parametri dei prodotti ricercati, poi viene mostrata la query composta in modo che l'utente mantenga il controllo delle ricerche che sta svolgendo ed infine vengono mostrati i risultati delle ricerche in forma tabellare. Il nome del prodotto è cliccabile in quanto è un link verso la pagina di descrizione del prodotto stesso.

prodotto.php: si tratta della pagina citata poc'anzi ed è la pagina più articolata dell'applicazione. Si occupa di visualizzare un'immagine del prodotto selezionato, i suoi dettagli e offre

la possibilità di selezionare le taglie/numeri e il sesso dei prodotti se si tratta di indumenti; un tasto a fondo descrizione permette di verificarne la quantità in base alle caratteristiche selezionate(indicate anche a fondo pagina per facilitare e velocizzare la navigazione dell'utente); infine un menù a tendina permette di selezionare la quantità desiderata da mettere nel carrello e un altro tasto di confermare la scelta fatta. Nel caso di attrezzature vale lo stesso discorso tranne per il fatto che non si possono selezionare le caratteristiche dei prodotti in quanto fissate per ogni articolo.

addcart.php: è la apagina che si occupa in pratica dell'aggiunta di un record nella tabella *Carrello* che lega l'utente al prodotto appena selezionato; in base all'esito dell'operazione. La pagina genera l'opportuna segnalazione di successo o di fallimento totale o parziale in seguito ad un'eventuale modifica contemporanea della base di dati.

carrello.php: mostra i prodotti (e i relativi dettagli) che l'utente ha inserito nel carrello durante la navigazione, ordinati per prezzo in maniera decrescente; a fianco ad ogni prodotto un link permette di rimuovere un elemento dal carrello tramite una get che viene ricevuta dalla pagina stessa; a fondo pagina viene mostrato il totale dell'ordine ed eventualmente il totale scontato se l'utente possiede un buono sconto; in questa situazione viene anche data la possibilità di applicare o meno lo sconto al carrello; infine, a fondo pagina troviamo il bottone per confermare l'acquisto del carrello.

conferma_ordine.php: è la pagina che si occupa di raccogliere la richiesta, da parte dell'utente, di procedere all'acquisto dei prodotti nel carrello; qui tramite procedure e funzioni avviene la creazione del nuovo ordine, la copia dei prodotti dalla tabella *Prodotti* a *Prodotti_venduti* e le relative entry per la tabella *Prodotti_ordini*; di conseguenza a ciò avviene l'eliminazione dei prodotti acquistati dalla tabella *Prodotti* e di tutte le entry relative alle altre tabelle le cui chiavi esterne puntavano ai prodotti interessati (grazie alla proprietà ON DELETE CASCADE delle FK); inoltre, nel caso fosse stato applicato uno sconto, esso verrebbe azzerato; infine viene mostrata un messaggio di conferma dell'acquisto con relativo messaggio di sconto in base all'ammontare del prezzo totale dei prodotti acquistati.

sconti.php: permette di vedere il proprio sconto a disposizione oppure gli sconti di tutti gli utenti nel caso la pagina fosse visualizzata da Amministratori.

ordini.php: come per gli sconti si può osservare l'elenco dei propri ordini oppure quelli di tutti gli utenti se Amministratori; gli ordini hanno un ID cliccabile per visualizzare i dettagli dell'ordine.

dettaglio_ordine.php: qui appunto si può osservare in dettaglio la lista dei prodotti che compongono l'ordine selezionato.

gestisci_utenti.php: è una pagina visibile soltanto dagli Amministratori (in specifico Proprietari) in quanto, oltre che dare la possibilità di tenere d'occhio la lista dei Clienti iscritti al sito e il relativo sconto, permette di azzerare tale sconto a discrezione dell'Amministratore e, più radicalmente, di eliminare l'utente stesso, liberando così i prodotti che eventualmente aveva nel carrello e perdendo gli ordini da lui effettuati.

del.php: è la pagina che si occupa di compiere le operazioni sopra citate; essa non genera alcina pagina in quanto reindirizza immediatamente a *utenti.php* per controllare che gli effetti siano corretti.

login.php: permette semplicemente di effettuare l'accesso al sito creando la sessione per l'utente sul server, indispensabile in ogni pagina per regolare il comportamento delle stesse in base al tipo di utente e le informazioni alle quali può avere accesso.

logout.php: si limita semplicemente a distruggere la sessione dell'utente.

register.php: permette di registrarsi al sito per poter effettuarne l'accesso; viene automaticamente creato uno sconto impostato a 0 associato all'utente appena inserito.

7 Conclusione

L'applicazione permette di effettuare tutte quelle che sono le operazioni principali che si possono effettuare su una base di dati (insermento, cancellazione, modifca). La gestione del mantenimento dello stato tra le pagine ha superato tutti i test. Lo stesso si può dire per funzioni, procedure e trigger a supporto della base di dati. Inoltre semplici funzioni JavaScript sono state realizzate a supporto dell'applicazione.

Al progetto entrambi i membri del gruppo hanno partecipato in maniera vicendevole e impegnata, comunicando assiduamente e in maniera molto serena con l'obiettivo di dimostrare quanto appreso durante il corso di Basi di dati e cercare di applicarlo al meglio.