UNIVERSITÀ DEGLI STUDI DI SALERNO

FACOLTÀ' DI SCIENZE MM. FF. NN.

CORSO DI LAUREA TRIENNALE IN INFORMATICA

Corso di Basi di Dati

Compra E Vinci

ANNO ACCADEMICO 2017/2018

Coordinatore del progetto:

Nome
Giuseppe Polese

Partecipanti:

Nome	Matricola
Bruno D’Agostino	0512103598
Raffaele Florio	0512103766

Indice

Compra e Vinci - Descrizione

Il progetto modella un database, usato da un operatore attraverso un'applicazione Java/JDBC, di un ecommerce che ai primi 100 classificati del mese regala un premio.

Per poter accedere alla classifica premi bisogna aver speso una quantità minima di denaro e i premi vengono divisi per fascia di spesa. Maggiore è la spesa, migliore è il premio.

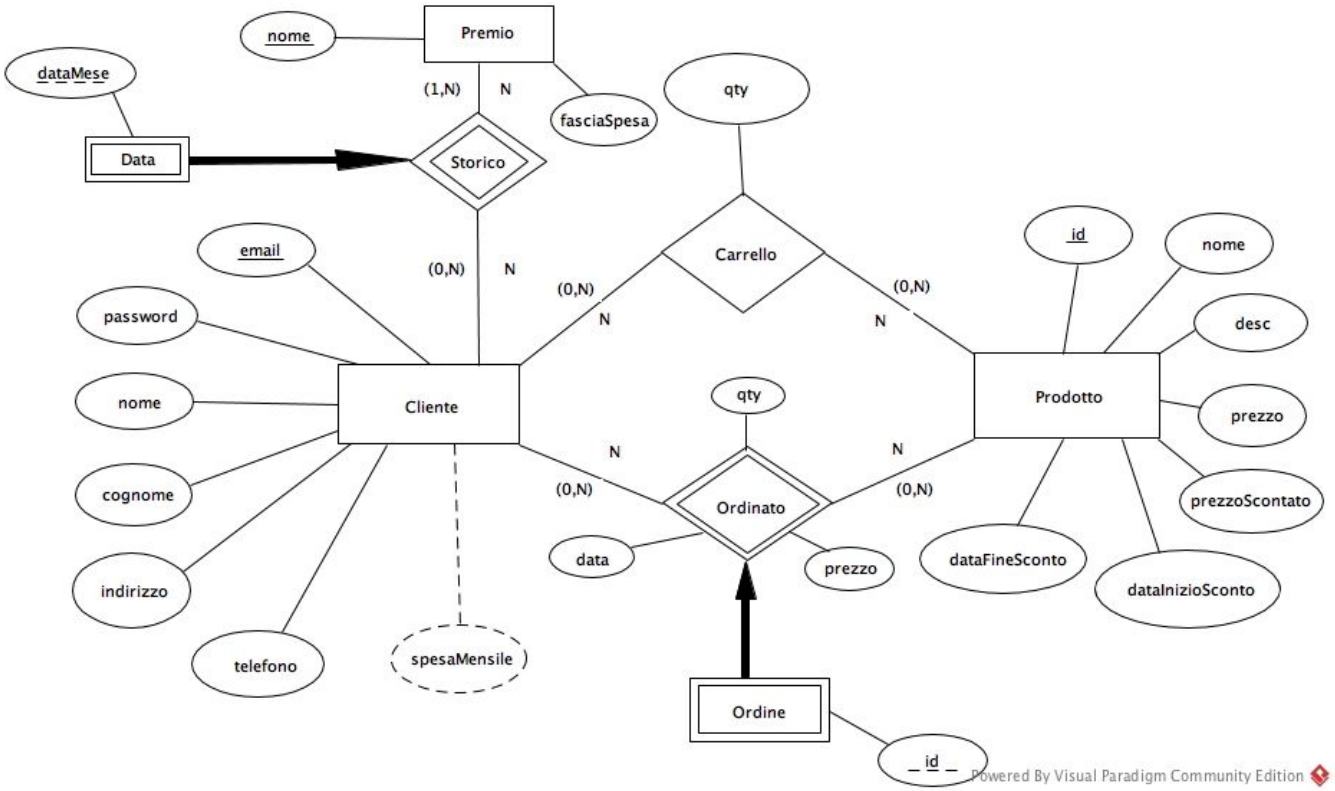
L'operatore ha la capacità di effettuare le operazioni di visualizzazione e modifica del database e relativo reset della classifica, possibile solo all'ultimo giorno del mese.

Le operazioni effettuabili dall'operatore sono:

- Visualizzazione ordini
- Visualizzazione clienti
- ***Inserimento clienti (OP1)***
- Visualizzazione classifica attuale
- Visualizzazione prodotti
- Visualizzazione prodotti più venduti del mese
- Visualizzazione carrello
- Visualizzazione sconti attivi
- Visualizzazione posizione di un cliente in classifica
- Visualizzazione premi
- Visualizzazione storico vincite
- ***Visualizzazione classifica vincitori e relativo reset (OP3)***
- ***Inserimento ordine (OP2)***

Le operazioni più frequenti sono quelle evidenziate in grassetto e in corsivo.

Schema Concettuale Entity-Relationship (ER) della Base di Dati



- Entita': Cliente
  - Descrizione: Persona che acquista prodotti.
  - Attributi: Email, Password, Nome, Cognome, Indirizzo, Telefono, Spesa Mensile.
    - Email: Indirizzo di posta elettronica, il dominio e' varchar.
    - Password: Parola d'ordine, il dominio e' varchar.
    - Nome: Nome del cliente. Il dominio e' varchar.
    - Cognome: Cognome del cliente. Il dominio e' varchar.
    - Indirizzo: Indirizzo di abitazione del cliente. Il dominio e' varchar.
    - Telefono: Numero di telefono del cliente. Il dominio e' varchar.
    - Spesa Mensile: Totale speso in un mese. Il dominio e' decimal.
  - Identificatori: Email.
    - Email: Presente come chiave esterna nelle relazioni Carrello, Ordinato e Storico.

- Entita': Prodotto
    - Descrizione: Prodotto acquistato dal cliente.
    - Attributi: ID, Nome, Desc, Prezzo, Prezzo Scontato, Data Inizio Sconto, Data Fine Sconto.
        - ID: Identificatore dell'entita' Prodotto. Il dominio e' varchar.
        - Nome: Nome del prodotto. Il dominio e' varchar.
        - Desc: Descrizione del prodotto. Il dominio e' varchar.
        - Prezzo: Prezzo del prodotto. Il dominio e' decimal.
        - Prezzo Scontato: Prezzo del prodotto dopo aver applicato lo sconto. Il dominio e' decimal.
        - Data Inizio Sconto: Data di quando viene applicato lo sconto. Il dominio e' date.
        - Data Fine Sconto: Data di quando viene rimosso lo sconto. Il dominio e' date.
    - Identificatori: ID.
        - ID: Presente come chiave esterna nelle relazioni Carrello e Ordinato.

- Relazione: Carrello
    - Descrizione: Contiene tutti i prodotti che acquista un cliente.
    - Attributi: QTY.
        - QTY: Quantita' di prodotti inseriti nel carrello. Il dominio e' int.
    - Cardinalita': N ←→ N. Perche' un cliente piu' avere N prodotti nel carrello e ogni prodotto puo' essere presente in N carrelli di clienti diversi.

- Entita': Ordine
    - Descrizione: Ordine effettuato dal cliente. Entita' debole.
    - Attributi: ID.
        - ID: Identificatore parziale dell'entita' Ordine. Il dominio e' int.
    - Identificatori: ID.

- Relazione: Ordinato
    - Descrizione: Cio' che viene ordinato dal cliente.
    - Attributi: QTY, Data, Prezzo.
        - QTY: Quantita' di prodotti ordinati. Il dominio e' int.

        - Data: Data di quando viene effettuato l'ordine. Il dominio e' date.
        - Prezzo: Prezzo pagato per l'ordine. Il dominio e' decimal.
    - Cardinalita': N ←→ N. Perche' un cliente puo' aver ordinato N prodotti e ogni prodotto puo' essere presente in N ordini di clienti diversi.

- Entita' Premio
    - Descrizione: Premio vinto dai clienti che fanno acquisti.
    - Attributi: Nome, Fascia Spesa.
        - Nome: Identificatore del premio. Il dominio e' varchar.
        - Fascia Spesa: Costo spesa effettuata per aver vinto il premio. Il dominio e' decimal.
    - Identificatori: Nome.
        - Nome: Presente come chiave esterna nella relazione Storico.

- Relazione: Storico
    - Descrizione: Contiene lo storico dei premi vinti dai clienti.
    - Cardinalita': N ←→ N. Perche' un cliente puo' avere N premi nel suo storico e ogni premio puo' essere presente in N storici di clienti diversi.

- Entita': Data
    - Descrizione: Data dello storico salvato. Entita' debole.
    - Attributi: Data Mese.
        - Data Mese: Identificatore parziale dell'entita' Data. Il dominio e' date.
    - Identificatori: Data Mese.

Carico Applicativo

Nel database vengono mantenuti i dati dell'ultimo anno, durante il quale solitamente vengono registrati circa 1000 clienti, i quali effettuano circa due ordini ciascuno (circa 2000 ordini). Inoltre vengono messi in vendita circa 5000 prodotti diversi. In media, in un anno, vengono messi a disposizione circa 1000 premi che i clienti vincono effettuando spese.

Tavola Volumi

CONCETTO	TIPO	VOLUME
Cliente	Entita'	1000
Carrello	Relazione	5000
Prodotto	Entita'	5000
Ordinato	Relazione	2000
Ordine	Entita'	2000
Storico	Relazione	1000
Premio	Entita'	1000
Data	Entita'	1000

Tavola Operazioni

- OP1: Inserisci cliente.
- OP2: Inserimento ordine.
- OP3: Report mensile con i dati di ciascun cliente, compreso il valore della spesa mensile.

OPERAZIONE	TIPO	FREQUENZA
OP1	I	83 volte/mese
OP2	I	166 volte/mese
OP3	B	1 volta/mese

Tavole Accessi

OP1

CONCETTO	COSTRUTTO	ACCESSI	TIPO
Cliente	Entita’	1	S

OP2 con ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
Ordine	Entita’	1	S
Ordinato	Relazione	1	L
Cliente	Entita’	1	L

Cliente	Entita’	1	S

OP2 senza ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
Ordine	Entita’	1	S
Ordinato	Relazione	1	L

OP3 con ridondanza

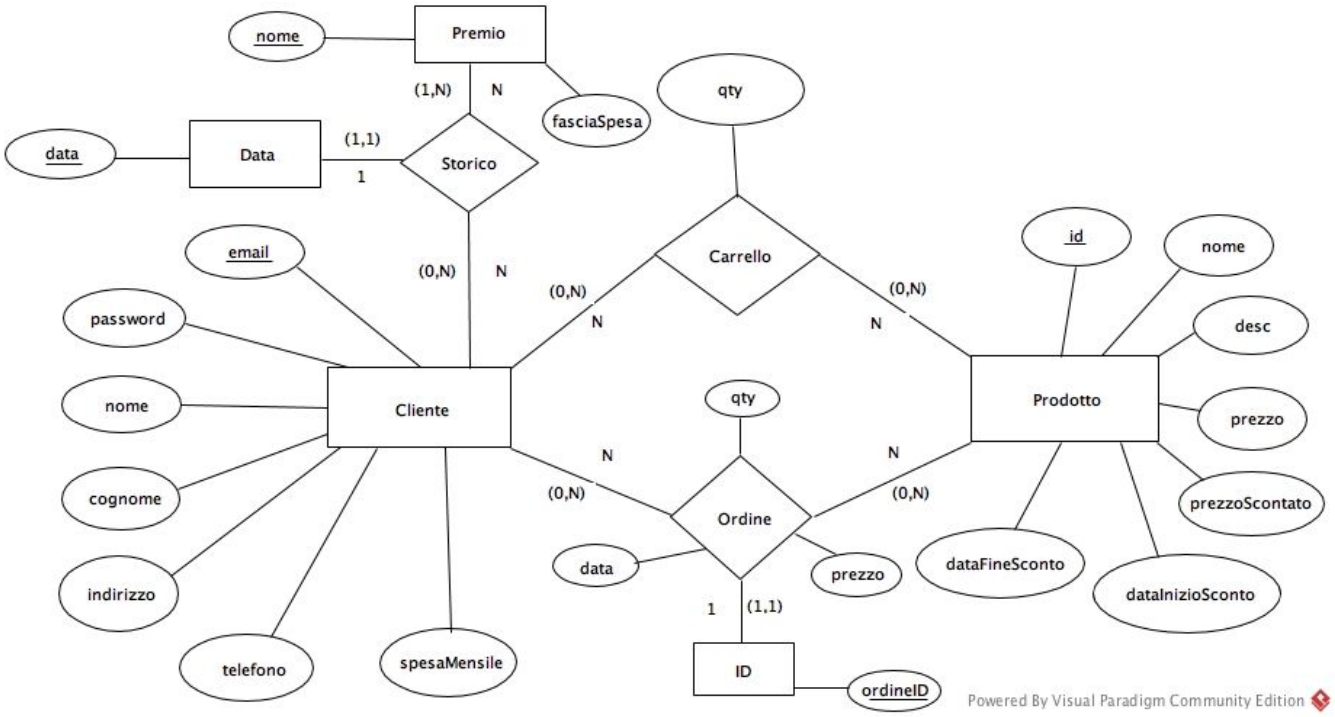
CONCETTO	COSTRUTTO	ACCESSI	TIPO
Cliente	Entita’	1	L

OP3 senza ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
Cliente	Entita’	1	L
Ordinato	Relazione	2	L

Dopo aver eseguito le analisi riguardo le operazioni piu’ frequenti da effettuare sulla base di dati, si e’ giunti alla conclusione di non eliminare l’attributo ridondante “spesaMensile” dell’entita’ Cliente perche’ altrimenti l’operazione 2, ma soprattutto l’operazione 3, avrebbe avuto un costo troppo elevato. Il costo dell’operazione 2 senza ridondanza e’ di 498 ed il costo dell’operazione 3, sempre senza ridondanza, e’ di 3000, per un totale di 3498. Invece, con ridondanza, l’operazione 2 ha un costo di 996 e l’operazione 3 ha un costo di 1000, per un totale di 1996.

Schema ER Ristrutturato



- L’attributo ridondante ‘spesaMensile’ non e’ stato cancellato a causa di efficienza quando si esegue l’operazione OP3.
- E’ stata cancellata la relazione ‘Ordinato’ e l’entita’ debole ‘Ordine’ e’ diventata relazione tra Cliente e Prodotto e la sua chiave primaria sara’ la combinazione tra ‘id’, ‘email_cliente’ e ‘id_prodotto’. Inoltre ha ereditato tutti gli attributi della vecchia relazione ‘Ordinato’.
- E’ stata cancellata l’entita’ debole ‘Data’ ed il suo attributo e’ stato assegnato alla relazione ‘Storico’. L’attributo ‘data’ della relazione ‘Storico’ sara’ chiave primaria di ‘Storico’ insieme alla chiave ‘email_cliente’ dell’entita’ Cliente.

Schema Logico e Statement DDL

Cliente(email, password, nome, cognome, indirizzo, telefono, spesaMensile);

```
CREATE TABLE Cliente (
  email varchar(200) NOT NULL,
  password char(60) NOT NULL,
  nome varchar(30) NOT NULL,
  cognome varchar(30) NOT NULL,
  indirizzo varchar(100) NOT NULL,
  telefono varchar(14) NOT NULL,
  spesaMensile decimal(8,2) unsigned NOT NULL DEFAULT '0.00',
  PRIMARY KEY (email)

)
```

Prodotto(id, nome, desc, prezzo, prezzoScontato, dataInizioSconto, dataFineSconto);

```
CREATE TABLE Prodotto (
  id varchar(15) NOT NULL,
  nome varchar(50) NOT NULL,
  desc varchar(255) DEFAULT NULL,
  prezzo decimal(8,2) unsigned NOT NULL,
  prezzoScontato decimal(8,2) unsigned DEFAULT NULL,
  dataInizioSconto date DEFAULT NULL,
  dataFineSconto date DEFAULT NULL,
  PRIMARY KEY (id)

)
```

Premio(nome, fasciaSpesa);

```
CREATE TABLE Premio (
  nome varchar(200) NOT NULL,
  fasciaSpesa decimal(8,2) unsigned NOT NULL,
  PRIMARY KEY (nome)
)
```

Ordine(id, email_cliente, id_prodotto, prezzo, qty, data);

```
CREATE TABLE Ordine (
  id int(11) NOT NULL,
  email_cliente varchar(200) NOT NULL,
  id_prodotto varchar(15) NOT NULL,
  prezzo decimal(8,2) unsigned NOT NULL,
  qty int(10) unsigned NOT NULL,
  data date NOT NULL,
  PRIMARY KEY (id,email_cliente,id_prodotto),
  CONSTRAINT fk_Ordine_Cliente FOREIGN KEY (email_cliente) REFERENCES
Cliente (email) ON DELETE CASCADE

  ON UPDATE CASCADE,
  CONSTRAINT fk_Ordine_Prodotto FOREIGN KEY (id_prodotto) REFERENCES
Prodotto (id) ON DELETE NO ACTION ON UPDATE CASCADE
)
```

Storico(data, email_cliente, nomePremio);

```
CREATE TABLE Storico (
  data date NOT NULL,
  email_cliente varchar(200) NOT NULL,
  nomePremio varchar(45) NOT NULL,
  PRIMARY KEY (data,email_cliente),
  CONSTRAINT fk_Storico_Cliente FOREIGN KEY (email_cliente) REFERENCES
Cliente (email) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT fk_Storico_Premio FOREIGN KEY (nomePremio) REFERENCES Premio
(nome) ON DELETE

  CASCADE ON UPDATE CASCADE
)
```

Carrello(email_cliente, id_prodotto, qty);

```
CREATE TABLE Carrello (
  email_cliente varchar(200) NOT NULL,
  id_prodotto varchar(15) NOT NULL,
  qty int(10) unsigned NOT NULL,
  PRIMARY KEY (email_cliente,id_prodotto),
  CONSTRAINT fk_Carrello_Cliente FOREIGN KEY (email_cliente) REFERENCES
Cliente (email) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT fk_Carrello_Prodotto FOREIGN KEY (id_prodotto) REFERENCES
Prodotto (id) ON DELETE CASCADE ON UPDATE CASCADE
)
```

Schema Esterno

Attraverso questa vista è possibile ottenere tutte le tuple dei clienti in classifica e i relativi premi assegnati:

```
CREATE VIEW ClassificaClientiV AS
SELECT C.email, C.password, C.nome, C.cognome, C.indirizzo, C.telefono,
C.spesaMensile,
P.nome as nomePremio
FROM Cliente C, Premio P
WHERE
C.spesaMensile >  50 AND
C.spesaMensile >= (SELECT MAX(fasciaSpesa) FROM Premio P2 WHERE
P2.fasciaSpesa <= C.spesaMensile) AND

P.fasciaSpesa = (SELECT MAX(fasciaSpesa) FROM Premio P3 WHERE P3.fasciaSpesa
<=
C.spesaMensile)
ORDER BY C.spesaMensile DESC;
```

Attraverso questa vista è possibile ottenere i prodotti più venduti del mese corrente:

```
CREATE VIEW ProdottiPiuVendutiV AS
SELECT *
FROM Prodotto P
JOIN

(
SELECT id_prodotto, SUM(Ordine.qty) AS nOrdinati
FROM Ordine
WHERE MONTH(data) = MONTH(CURDATE())
GROUP BY id_prodotto
) AS idPPiuvenduti

ON id_prodotto = P.id ORDER BY nOrdinati DESC;
```

Attraverso questa vista è possibile ottenere i prodotti scontati:

```
CREATE VIEW ProdottiScontatiV AS
SELECT *
FROM Prodotto
WHERE dataInizioSconto <= CURDATE() AND dataFineSconto >= CURDATE()
ORDER BY dataFineSconto ASC
```

Implementazione della Base di Dati utilizzando MySQL

Istruzioni MySQL per creare la Base di Dati

Le istruzioni MySQL per creare la base di dati si trovano in un file a parte chiamato FINALE_CEV_SD.sql

Script SQL per popolare la Base di Dati

Lo script SQL per popolare la base di dati si trova in un file a parte chiamato QueryPopolamentoEDefViste.sql

Applicazione Java/JDBC

L’applicazione Java/JDBC si trova in una cartella a parte chiamata Progetto.