SPRITZ
SECURITY & PRIVACY
RESEARCH GROUP

Basi di Dati

Progetto Basi di Dati 2014/15 (parte 2)

Prof. **Mauro Conti**

Dipartimento di Matematica - Università degli studi di Padova

conti@math.unipd.it - http://www.math.unipd.it/~conti

Giuseppe Cascavilla, PhD student

sites.google.com/site/gcascavilla

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Quante?

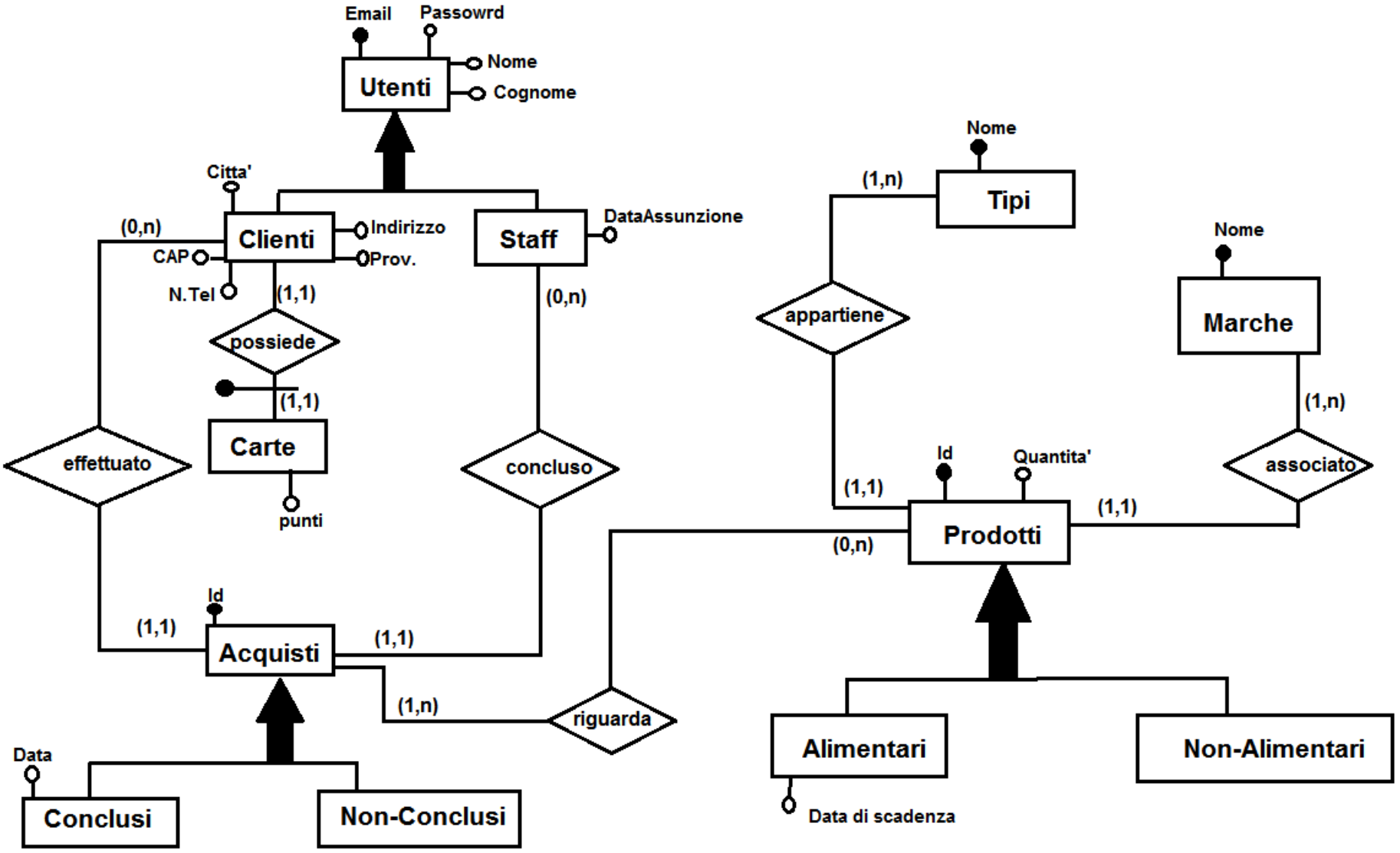
1. Il progetto deve includere

    a. Almeno 6 query significative
    b. Almeno 2 funzioni
    c. Almeno 2 trigger

Descrizione Query, Trigger e Funzioni

1. Per ogni query, trigger, funzione o procedura:
    a. Descrizione testuale
    b. Codice
    c. **Solo per le query: Output**
        i. L'output puo' essere relativo a uno stato della base di dati anche non corrispondente a quello nel momento della consegna

SuperMarket Online: E-R diagram



SuperMarket Online: schema relazionale

CLIENTE (**E-mail**, Password, Nome, Cognome, Citta', Indirizzo, Provincia, CAP, NumTel)

STAFF (**E-mail**, Password, Nome, Cognome, DataAssunzione)

CARTE (Punti, **Cliente**)

ACQUISTI (**Id,** Clienti**,** Staff, Stato, ConclusoInData)

AcquistiProdotti (**Acquisti, Prodotti**)

PRODOTTI (**Id**, Tipi, Marche, Prezzo, Nome, Quantita', Genere, DataScadenza)

TIPI (**Nome**)

MARCHE (**Nome**)

SuperMarket Online: query 1

Query che ritorna tutti gli id degli acquisti (conclusi e non) e le e-mail dei clienti che li hanno effettuati, nei quali sono stati acquistati solamente prodotti non alimentari ognuno dei quali in quantità maggiore di due pezzi.

SuperMarket Online: query 1


```
SELECT a.Id,a.Cliente
FROM Acquisti a
WHERE NOT EXISTS (
    SELECT *
    FROM AcquistiProdotti ap JOIN Prodotti p
        ON (ap.Prodotto=p.Id)
    WHERE a.Id=ap.Acquisto AND (Genere='alim' OR ap.Quantita <= 2)
);
```

SuperMarket Online: query 1


```
SELECT a.Id,a.Cliente
FROM Acquisti a
WHERE NOT EXISTS (
      SELECT *
      FROM AcquistiProdotti ap JOIN Prodotti p
            ON (ap.Prodotto=p.Id)
      WHERE a.Id=ap.Acquisto AND (Genere='alim' OR ap.Quantita <= 2)
);
```

In questa query si selezionano tutti gli acquisti nei quali non esiste un prodotto che è alimentare o un prodotto in quantità minore o uguale a due pezzi.

SuperMarket Online: query 2

Query che ritorna le e-mail dei clienti che non hanno effettuato nessun acquisto o quelli che in tutti i loro acquisti hanno comprato tutti prodotti dello stesso tipo (cioè non è possibile che su un acquisto ci siano due prodotti aventi tipo differente).

SuperMarket Online: query 2


```
SELECT c.Email
FROM Clienti c
WHERE c.Email NOT IN (
      SELECT a.Cliente
      FROM Acquisti a )
OR NOT EXISTS (
      SELECT *
      FROM Acquisti acq
      WHERE acq.Cliente=c.Email AND EXISTS (
            SELECT *
            FROM AcquistiProdotti ap JOIN Prodotti p ON (ap.Prodotto=p.Id)
            WHERE acq.Id=ap.AcquistoAND p.Tipo <> ANY (
                  SELECT p1.Tipo
                  FROM AcquistiProdotti ap1 JOIN Prodotti p1 ON (ap1.
            Prodotto=p1.Id)
                  WHERE ap.Acquisto=ap1.Acquisto
            )
      )
);
```

SuperMarket Online: query 2

In questa query si selezionano le e-mail dei clienti che non figurano nella colonna 'Cliente' della tabella Acquisti o le e-mail dei clienti per i quali non esiste un acquisto da loro effettuato in cui c'è almeno un prodotto che ha tipo diverso da un altro prodotto di quell'acquisto.

```
SELECT c.Email
FROM Clienti c
WHERE c.Email NOT IN (
        SELECT a.Cliente
        FROM Acquisti a )
OR NOT EXISTS (
        SELECT *
        FROM Acquisti acq
        WHERE acq.Cliente=c.Email AND EXISTS (
                SELECT *
                FROM AcquistiProdotti ap JOIN Prodotti p ON (ap.Prodotto=p.Id)
                WHERE acq.Id=ap.AcquistoAND p.Tipo <> ANY (
                        SELECT p1.Tipo
                        FROM AcquistiProdotti ap1 JOIN Prodotti p1 ON (ap1.Prodotto=p1.Id)
                        WHERE ap.Acquisto=ap1.Acquisto
                )
        )
);
```

SuperMarket Online: query 3

Query che restituisce l'id, il nome e la marca dei prodotti per i quali nel mese scorso è stato venduto un numero di pezzi complessivo superiore a 10 (vengono considerati solo gli acquisti conclusi).

SuperMarket Online: query 3


```
SELECT p.Id,p.Nome,p.Marca
FROM Prodotti p
WHERE p.Id IN (

SELECT p1.Id
    FROM Acquisti a JOIN AcquistiProdotti ap ON (a.Id=ap.Acquisto) JOIN
    Prodotti p1 ON (ap.Prodotto=p1.Id)

    WHERE Stato='concluso' AND EXTRACT(YEAR FROM
        a.ConclusoInData)=EXTRACT(YEAR
    FROM DATE_SUB(CURDATE(),INTERVAL 1
        MONTH)) AND EXTRACT(MONTH
    FROM a.ConclusoInData)=EXTRACT(MONTH
    FROM DATE_SUB(CURDATE(),INTERVAL 1
        MONTH))
    GROUP BY p1.Id
    HAVING SUM(ap.Quantita) > 10 );
```

SuperMarket Online: query 3

Nella query precedente vengono usate tre funzioni di MySQL che permettono la gestione delle date.

In particolare è stata usata la funzione CURDATE() che ritorna la data corrente nel formato 'YYYY-MM-DD', la funzione DATE_SUB che sottrae ad una certa data un intervallo di tempo e la funzione EXTRACT che estrae una parte di una data.

In tal caso, siccome si considerano gli acquisti effettuati nel mese scorso (dalla data corrente), si eseguono due operazioni:

- **si sottrae dalla data corrente un intervallo di tempo pari a un mese e si estrae da tutto ciò l'anno;**
- **si sottrae dalla data corrente un intervallo di tempo pari a un mese e si estrae da tutto ciò il mese.**

Perché un acquisto sia stato concluso nel mese passato deve essere che l'anno e il mese estratti dall'attributo 'ConclusoInData' sono uguali all'anno e mese estratti dalla data corrente meno l'intervallo di un mese.

SuperMarket Online: query 4

Query che restituisce il numero di prodotti (alimentari e non) che sono presenti (in una o più quantità) esattamente in un acquisto. Tale query è stata creata sfruttando anche una vista.

SuperMarket Online: query 4


```
DROP VIEW IF EXISTS
      ProdottiUnSoloAcquisto;
CREATE VIEW ProdottiUnSoloAcquisto AS
(
SELECT COUNT(*) AS NumProdotti
FROM Acquisti a JOIN AcquistiProdotti ap ON (a.Id=ap.Acquisto) JOIN Prodotti p
ON (ap.Prodotto=p.Id)
WHERE p.Id IN (
      SELECT p1.Id
      FROM AcquistiProdotti ap1 JOIN Prodotti p1 ON (ap1.Prodotto=p1.Id)
      WHERE a.Id=ap1.Acquisto
      )
      AND NOT EXISTS (
      SELECT *
      FROM AcquistiProdotti ap2 JOIN Prodotti p2 ON (ap2.Prodotto=p2.Id)
      WHERE a.Id <> ap2.Acquisto AND p.Id=p2.Id
      )
);

SELECT *
FROM ProdottiUnSoloAcquisto;
```

SuperMarket Online: query 4

Nella vista creata si contano tutti i prodotti che sono presenti in almeno un acquisto (attraverso l'operatore per la quantificazione esistenziale IN) e che non compaiono in nessun altro acquisto (mediante l'operatore per la quantificazione universale NOT EXISTS).

SuperMarket Online: query 5

Query che per ogni cliente che ha effettuato almeno un acquisto restituisce il costo totale massimo tra i totali dei suoi acquisti (senza considerare un eventuale sconto), mentre per i clienti che non hanno effettuato alcun acquisto ritorna 0 come costo totale massimo.

SuperMarket Online: query 5


```
SELECT a.Cliente,PrezzoTotaleAcquisto(a.Id) AS Costo_massimo
FROM Acquisti a JOIN AcquistiProdotti ap ON (a.Id=ap.Acquisto) JOIN Prodotti p
ON (ap.Prodotto=p.Id)
WHERE PrezzoTotaleAcquisto(a.Id) >= ALL
(
    SELECT PrezzoTotaleAcquisto(a1.Id)
    FROM Acquisti a1 JOIN AcquistiProdotti ap1 ON (a1.Id=ap1.Acquisto) JOIN
    Prodotti p1 ON (ap1.Prodotto=p1.Id)
    WHERE a.Id<>a1.Id AND a.Cliente=a1.Cliente
    )
        GROUP BY a.Cliente
        UNION
            SELECT c.Email,0 AS Costo_massimo
            FROM Clienti c
            WHERE c.Email NOT IN ( SELECT acq.Cliente
                                   FROM Acquisti acq
                                   )
            GROUP BY c.Email;
```

SuperMarket Online: query 5

Con la prima SELECT di questa query si seleziona per ogni cliente che ha effettuato almeno un acquisto (tali che la loro e-mail compaia come attributo 'Cliente' dell'acquisto) l'acquisto il cui prezzo totale è maggiore o uguale a tutti gli altri totali degli acquisti da loro fatti. In questa select, oltre a sfruttare la funzione “PrezzoTotaleAcquisto” per determinare il costo totale dell'acquisto, si usa anche l'operatore di quantificazione universale ALL.

Con la seconda SELECT, invece, si selezionano i clienti la cui e-mail non compare nella colonna 'Cliente' della tabella Acquisti e per questi viene ritornato 0 come costo totale massimo.

SuperMarket Online: funzione 1

Funzione che verifica che un certo prodotto di id "Id_prod" è disponibile in relazione alla quantità che il cliente desidera, tenendo in considerazione anche che non sia scaduto (se il prodotto è di genere alimentare).

```
CREATE FUNCTION ControllaDisponibilitaPezzi (Id_prod MEDIUMINT UNSIGNED, Qta SMALLINT
UNSIGNED)
RETURNS BOOL
BEGIN
DECLARE PezziDisponibili SMALLINT UNSIGNED;
DECLARE Gen ENUM('alim','non_alim');
DECLARE DataProdotto DATE;
DECLARE Disponibilita BOOL;
SELECT Quantita,Genere,DataScadenza INTO PezziDisponibili,Gen,DataProdotto
FROM Prodotti
WHERE Id=Id_prod;
IF(Qta <= PezziDisponibili AND (Gen='non_alim' OR (Gen='alim' AND DataProdotto > CURDATE()) ) )
THEN
        SET Disponibilita = 1;
ELSE
        SET Disponibilita = 0;
END IF;
RETURN Disponibilita;
END $
```

SuperMarket Online: funzione 2

Funzione che determina il prezzo totale di un certo acquisto di id "Id_acq“ eseguendo la somma di tutti i subtotali dati dal prezzo unitario del prodotto per il numero di pezzi che presenta.

```
CREATE FUNCTION PrezzoTotaleAcquisto (Id_acq MEDIUMINT
UNSIGNED)
RETURNS DECIMAL(6,2)

BEGIN
DECLARE Totale DECIMAL(6,2);
SELECT SUM(ap.Quantita*p.Prezzo) INTO Totale
FROM Acquisti a JOIN AcquistiProdotti ap ON(a.Id=ap.Acquisto)
     JOIN Prodotti p ON(ap.Prodotto=p.Id)
WHERE a.Id=Id_acq;

RETURN Totale;
END $
```

SuperMarket Online: funzione 3

Università degli Studi di Padova

Funzione che restituisce il prezzo del prodotto avente costo minimo oppure il prezzo del prodotto avente costo massimo a seconda se in input viene passato il valore 'min' o 'max'.

```
CREATE FUNCTION PrezzoMinMaxProdotti(MinMax ENUM('min','max'))
RETURNS DECIMAL(4,2)
BEGIN
DECLARE Price DECIMAL(4,2);
      IF(MinMax='min')
      THEN
SELECT min(Prezzo) INTO Price
FROM Prodotti
WHERE (Genere='non_alim' OR (Genere='alim' AND DataScadenza>CURDATE()) );
ELSE /* MinMax='max' */
      SELECT max(Prezzo) INTO Price
      FROM Prodotti
      WHERE (Genere='non_alim' OR (Genere='alim' AND DataScadenza>CURDATE())
);
END IF;
RETURN Price;
END $
```

SuperMarket Online: funzione 4

Funzione che ritorna il numero di punti presenti nella carta di un certo cliente la cui email viene data in input.

```
CREATE FUNCTION PuntiCarta (Email_cliente VARCHAR(255))
RETURNS SMALLINT UNSIGNED
BEGIN

DECLARE Pti SMALLINT UNSIGNED;
SELECT Punti INTO Pti
FROM Carte
WHERE Cliente=Email_cliente;

RETURN Pti;
END $
```

SuperMarket Online: funzione 5

Funzione che determina il numero di punti derivanti dagli acquisti in fase di lavorazione di un certo cliente la cui email viene data in input (tale funzione sfrutta anche un cursore). Tale funzione è utile nell'interfaccia web quando un cliente, che sta effettuando un acquisto, visualizza il carrello. Se il totale è superiore a 50 o a 100 euro, può avere diritto ad uno sconto. Questo succede quando i punti nella carta del cliente sono maggiore o uguale a 15. Se, però, il cliente ha altri ordini in lavorazione, allora, per verificare se l'acquisto che sta per fare (se superiore a 50 o 100 euro) può avere diritto allo sconto, si tengono conto dei punti totali (quelli già nella carta più quelli che potrebbero venire aggiunti se tutti i suoi acquisti in lavorazione verrebbero conclusi). In seguito, al momento della conclusione dei vari acquisti, si verifica se effettivamente lo sconto su un certo acquisto può essere applicato verificando se il numero di punti nella carta del cliente è maggiore o uguale a 15.

SuperMarket Online: funzione 5


```
CREATE FUNCTION
ContaPuntiInLavorazione (Cli
VARCHAR(255))
RETURNS SMALLINT UNSIGNED
BEGIN
DECLARE Pti SMALLINT UNSIGNED
DEFAULT 0;
DECLARE Id_acq MEDIUMINT
UNSIGNED;
DECLARE SommaPti SMALLINT
UNSIGNED DEFAULT 0;
DECLARE Done INT DEFAULT 0;
DECLARE Cur_acq CURSOR FOR
      SELECT Id,PuntiAcquisto
      FROM Acquisti
      WHERE Stato='non_concluso';

DECLARE CONTINUE HANDLER FOR
NOT FOUND
      SET Done=1;

OPEN Cur_acq;
REPEAT
      FETCH Cur_acq INTO Id_acq,Pti;
      IF NOT Done THEN
            IF( ((PuntiCarta(Cli)+SommaPti) >=
            15) AND (PrezzoTotaleAcquisto
            (Id_acq)>100 OR
            PrezzoTotaleAcquisto(Id_acq)>50) )
            THEN
            SET SommaPti=SommaPti+Pti-15;
            ELSE
                  SET SommaPti=SommaPti+Pti;
            END IF;
      END IF;
UNTIL Done END REPEAT;
CLOSE Cur_acq;
RETURN SommaPti;
END $
```

SuperMarket Online: procedure 1

Procedura che restituisce gli acquisti nei quali è stato comprato un numero di pezzi di prodotti non alimentari superiore o uguale al numero dato in input.

```
CREATE PROCEDURE NumeroPezzi (IN numero MEDIUMINT UNSIGNED)
BEGIN
    SELECT a.Id AS CodiceAcquisto, SUM(ap.Quantita) AS
NumeroPezziProdNonAlim
    FROM Acquisti a JOIN AcquistiProdotti ap ON(a.Id=ap.Acquisto) JOIN
Prodotti p
        ON(ap.Prodotto=p.Id)
    WHERE p.Genere='non_alim'
    GROUP BY a.Id
    HAVING SUM(ap.Quantita) >= numero;
END $
```

SuperMarket Online: procedure 2

Procedura che ritorna il numero di pezzi disponibili per ogni tipo presente nel database. Con la prima SELECT si ottiene il numero di pezzi disponibili per i prodotti non alimentari o per quelli alimentari non scaduti. Con la seconda SELECT si ottengono tutti i tipi per i quali non esiste un prodotto alimentare di quel tipo che non sia scaduto (ossia che sia disponibile). Quindi, tale select seleziona tutti i tipi che presentano tutti prodotti alimentari scaduti (dunque 0 pezzi disponibili per questi tipi).

SuperMarket Online: procedure 2


```
CREATE PROCEDURE PezziDisponibili()
BEGIN

SELECT p.Tipo, SUM(p.Quantita) AS Pezzi_Disponibili
FROM Prodotti p
WHERE p.Genere='non_alim' OR (p.Genere='alim' AND p.DataScadenza > CURDATE())
GROUP BY p.Tipo

UNION

SELECT p.Tipo, 0 AS Pezzi_Disponibili
FROM Prodotti p
WHERE p.Genere='alim' AND NOT EXISTS (
      SELECT *
      FROM Prodotti p1
      WHERE p1.Genere=p.Genere AND p1.DataScadenza > CURDATE() AND p.Tipo=p1.
Tipo
            )
GROUP BY p.Tipo;
END $
```

SuperMarket Online: trigger 1

Trigger che dopo l'inserimento nel database di un nuovo cliente crea un record nella tabella Carte. In tal modo ad ogni cliente viene associata una “carta del negozio” avente all'inizio un numero di punti pari a 0 per default.

```
CREATE TRIGGER CreaCartaCliente
AFTER INSERT ON Clienti
FOR EACH ROW

BEGIN

INSERT INTO Carte (Cliente) VALUE (NEW.Email);
END $
```

SuperMarket Online: trigger 2

Trigger che permette di aggiungere i punti che derivano da un acquisto nella carta del cliente che lo ha effettuato. Questo succede dopo aver aggiornato la tabella Acquisti ponendo come 'concluso' un ordine. Il trigger verifica se l'acquisto ha diritto allo sconto (se in quel momento nella carta ci sono punti sufficienti e se il costo totale supera i 100 o 50 euro).

Se lo sconto può essere applicato allora dai punti nella carta si aggiungono quelli dell'acquisto appena concluso e al contempo se ne tolgono 15, altrimenti si aggiungono solo i punti derivanti dall'acquisto appena concluso.

SuperMarket Online: trigger 2


```
CREATE TRIGGER AggiungiPunti
AFTER UPDATE ON Acquisti
FOR EACH ROW

BEGIN
    IF( PuntiCarta(OLD.Cliente)>=15 AND (PrezzoTotaleAcquisto(OLD.Id)>100
OR
        PrezzoTotaleAcquisto(OLD.Id)>50) )
    THEN
        UPDATE Carte
        SET Punti=Punti+NEW.PuntiAcquisto-15
        WHERE Cliente=OLD.Cliente;
    ELSE
        UPDATE Carte
        SET Punti=Punti+NEW.PuntiAcquisto
        WHERE Cliente=OLD.Cliente;
    END IF;
END $
```

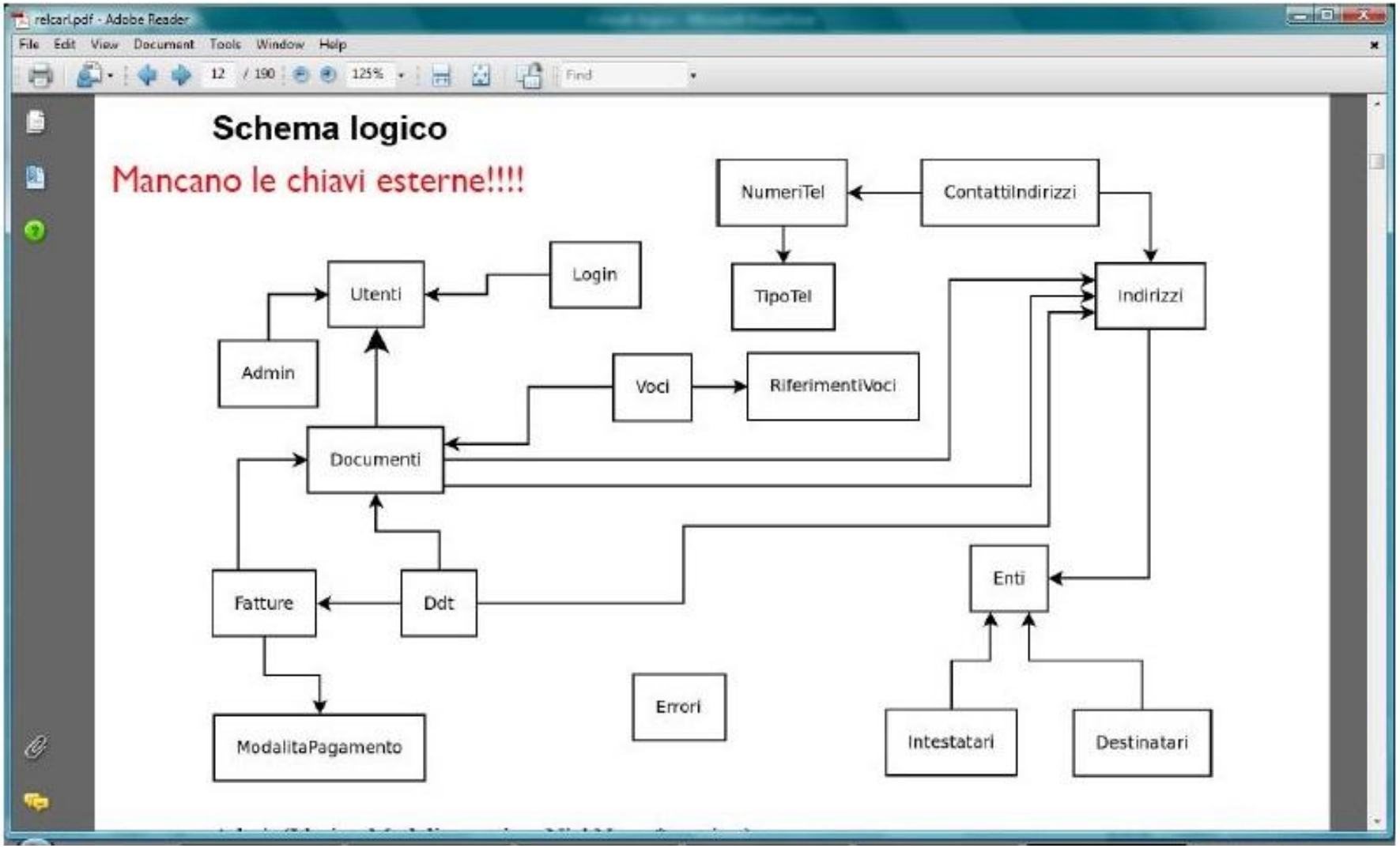
SuperMarket Online: trigger 3

Trigger che impedisce di concludere un acquisto in lavorazione di un certo cliente se quest'ultimo ne ha altri di precedenti in fase di lavorazione.

```
CREATE TRIGGER ConcludiAcquisti
BEFORE UPDATE ON Acquisti
FOR EACH ROW

BEGIN
    IF(AcquistiPrecedenti(OLD.Id,OLD.Cliente) > 0)
    THEN
        INSERT INTO Errori
        VALUE ('E" necessario prima concludere gli ordini precedenti del
    cliente');
    END IF;
END $
```

Document Manager: schema ERRATO



Document Manager: query 1


```
SELECT DISTINCT(e.Piva), e.Nome
FROM Enti e JOIN Indirizzi i ON (e.Piva=i.Ente)
WHERE i.Id IN (
      SELECT d.indPartenza
      FROM Documenti d
      WHERE year(d.Data)=$anno AND month(d.Data)=$mese AND d.Id IN (
            SELECT dd.NumeroDoc
            FROM Ddt dd
            WHERE isNull(dd.Fattura)
      )
)

OR e.Piva IN (SELECT Ente FROM Intestatari);
```

Document Manager: query 1

Descrizione: In questa interrogazioni sono presenti due variabili dipendenti dalla scelta dell'utente:

- $anno → rappresenta un anno (es. 2009, 2010, 2000 ecc);
- $mese → rappresenta un mese in valore numerico (es. 1 = gennaio, 2 = febbraio, ecc).

Questa interrogazione viene eseguita quando un utente sceglie di creare una nuova fattura nel gestionale. La richiesta al database è quella di fornire la partita iva di tutti gli enti che possono creare una fattura. Questi enti vengono scelti in base a due condizioni:

- Hanno almeno un documento di trasporto nel mese indicato da $mese e nell'anno $anno e quel Ddt non è ancora associato ad una fattura.

Questa condizione viene soddisfatto da questa parte della sql:

```
WHERE i.Id IN (SELECT d.indPartenza FROM Documenti d WHERE year(d.Data)=$anno AND month(d.Data)=$mese
```

- L'ente è un ente intestatario, quindi presente nella relazione “Intestatari”. Questo viene garantito dalla condizione:

```
OR e.Piva IN (SELECT Ente FROM Intestatari)
```

Non è stato sufficiente l'ultima condizione, in quanto un ente poteva essere intestatario e aver emesso dei documenti di trasporto, ma prima di creare la fattura per quei documenti essere stato tolto dalla lista degli intestatari e quindi con solo la seconda parte non sarebbe stato presente nei risultati della query.

Document Manager: query 2

```
DROP VIEW IF EXISTS
TotaleVociFatture;
CREATE VIEW TotaleVociFatture AS

SELECT (v.Qta*r.Prezzo) AS Importo,
      f.NumeroDoc AS Fattura, v.Id AS
      Voce
FROM (Fatture f JOIN Voci v ON
      (f.NumeroDoc=v.NumeroDoc))
      JOIN RiferimentiVoci r
      ON
(v.CodiceArt IS NOT NULL AND v.
AnnoArt IS NOT NULL
      AND v.CodiceArt=r.CodiceAND
v.AnnoArt=r.Anno)

UNION

SELECT (v.Qta*r.Prezzo) AS Importo,
d.Fattura AS Fattura, v.Id
      AS Voce
FROM (Ddt d JOIN Voci v ON (d.Fattura
      IS NOT NULL AND v.Stato='ddt' AND
      d.NumeroDoc=v.NumeroDoc))
JOIN RiferimentiVoci r ON
(v.CodiceArt IS NOT NULL AND
      v.AnnoArt IS NOT NULL AND v.
      CodiceArt=r.CodiceAND
      v.AnnoArt=r.Anno);

DROP VIEW IF EXISTS TotaleFatture;
CREATE VIEW TotaleFatture AS
SELECT sum(Importo) AS Totale, Fattura
FROM TotaleVociFatture
GROUP BY Fattura;
```

Document Manager: query 2

La query in esame permette di conoscere il totale di ogni singola fattura presente nel database.
Essa è divisa nelle seguenti 4 fasi:
1° fase:
Viene eseguita la seguente SELECT:

SELECT (v.Qta*r.Prezzo) AS Importo…

Questa SELECT calcola per ogni fattura il totale delle singole voci che sono direttamente appartenenti alla fattura.
Esempio:
Voce 1 → Id:1 – Qta: 2 – Prezzo 0.20 → totale: 0.40 –
IdFattura: 23
Voce 2 → Id:2 – Qta: 10 – Prezzo 2.40 → totale: 24 –
IdFattura: 33
…...
Voce N → Id:N – Qta: 13 – Prezzo 0.50 → totale: 6.50 –
IdFattura: 23
2° fase:
Viene eseguita la seguente SELECT:

SELECT (v.Qta*r.Prezzo) AS Importo,….

Questa SELECT effettua il calcolo di tutte le voci fatturabili (Stato voce → 'ddt') appartenenti ai vari documenti di trasporto che sono legati alla fattura in esame. Il risultato della SELECT è del tutto analogo a quello della fase 1.

Document Manager: query 2


```
DROP VIEW IF EXISTS TotaleVociFatture;
CREATE VIEW TotaleVociFatture AS
SELECT (v.Qta*r.Prezzo) AS Importo, f.NumeroDoc AS Fattura, v.Id AS Voce
FROM (Fatture f JOIN Voci v ON (f.NumeroDoc=v.NumeroDoc)) JOIN RiferimentiVoci r
ON (v.CodiceArt IS
      NOT NULL AND v.AnnoArt IS NOT NULL AND v.CodiceArt=r.CodiceAND v.
AnnoArt=r.Anno)

UNION

SELECT (v.Qta*r.Prezzo) AS Importo, d.Fattura AS Fattura, v.Id AS Voce
FROM (Ddt d JOIN Voci v ON (d.Fattura IS NOT NULL AND v.Stato='ddt' AND d.
NumeroDoc=v.NumeroDoc))
      JOIN RiferimentiVoci r ON (v.CodiceArt IS NOT NULL AND v.AnnoArt IS NOT
      NULL AND v.CodiceArt=r.CodiceAND v.AnnoArt=r.Anno);


DROP VIEW IF EXISTS TotaleFatture;
CREATE VIEW TotaleFatture AS
SELECT sum(Importo) AS Totale, Fattura
FROM TotaleVociFatture
GROUP BY Fattura;
```

Document Manager: query 2

3° fase:
Si esegue l'**UNION** delle fasi 1 e 2, quindi ora i due risultati vengono uniti e viene utilizzato per creare una vista (VIEW) di nome “TotaleVociFatture”

4° fase:
La quarta fase viene realizzata della seguente VIEW/SELECT:

DROP VIEW….

Essa raggruppa le voci della VIEW TotaleVociFatture in base al campo Fattura (identifica univocamente una singola fattura) e somma gli importi delle singole voci, infine crea la relativa VIEW.

Document Manager: funzione 1


```
DROP FUNCTION IF EXISTS NumeroDdt;
CREATE FUNCTION NumeroDdt(d DATE, intestatario CHAR(11)) RETURNS INT

BEGIN

    DECLARE num INT;
    SELECT max(Numero) Into num
    FROM ((Ddt dd JOIN Documenti do ON (dd.NumeroDoc=do.Id)) JOIN Indirizzi i ON (do.
    indPartenza=i.Id)) JOIN Enti e ON (i.Ente=e.Piva)
    WHERE e.Piva=intestatario AND year(do.Data)=year(d);

        IF num IS NULL
            THEN SET num = 1;
            ELSE SET num = num + 1;
        END IF;

RETURN num;
END
```

Descrizione:

- La funzione NumeroDdt permette di elaborare qual'è il successivo numero di D.d.t. che deve essere emesso secondo la data (d DATE) e l'intestatario (intestatario CHAR(11)) .

Document Manager: funzione 2


```
DROP FUNCTION IF EXISTS NumeroFattura;
CREATE FUNCTION NumeroFattura(d DATE, intestatario CHAR(11)) RETURNS INT
BEGIN
DECLARE num INT;
SELECT max(Numero) Into num
FROM ((Fatture dd JOIN Documenti do ON (dd.NumeroDoc=do.Id)) JOIN Indirizzi i
ON (do.indPartenza=i.Id))
      JOIN Enti e ON (i.Ente=e.Piva)
WHERE e.Piva=intestatario AND year(do.Data)=year(d);
IF num IS NULL
      THEN SET num = 1;
      ELSE SET num = num + 1;
END IF;

RETURN num;
END
```

Descrizione:

- La funzione NumeroFattura permette di elaborare qual'è il successivo numero di Fattura che deve essere emesso secondo la data (d DATE) e l'intestatario (intestatario CHAR(11)) .

Document Manager: funzione 3


```
DROP FUNCTION IF EXISTS PosizioneDisponibile;
CREATE FUNCTION PosizioneDisponibile(d INT) RETURNS INT
BEGIN
    DECLARE num INT;
    SELECT max(Posizione) Into num
    FROM Voci
    WHERE NumeroDoc=d;

    IF num IS NULL
    THEN SET num = 1;
    ELSE SET num = num + 1;
    END IF;

RETURN num;
END
```

Descrizione:

- La funzione PosizioneDisponibile permette di elaborare qual'è la successiva posizione disponibile di una voce all'intero di un documento per permettere all'utente di dare un ordine personalizzato alle voci.

Document Manager: trigger 1


```
DROP TRIGGER IF EXISTS InsQuantitaVoci;
CREATE TRIGGER InsQuantitaVoci BEFORE INSERT ON Voci
FOR EACH ROW
BEGIN
      IF NEW.Qta < 0 THEN
      INSERT INTO Errori(chiave) VALUES (NULL);
      END IF;
END;

DROP TRIGGER IF EXISTS UpQuantitaVoci;
CREATE TRIGGER UpQuantitaVoci BEFORE UPDATE ON Voci
FOR EACH ROW
BEGIN
      IF NEW.Qta < 0 THEN
      INSERT INTO Errori(chiave) VALUES (NULL);
      END IF;
END;
```

Descrizione:

- I trigger InsQuantitaVoci (interviene in caso di INSERT su Voci) e UpQuantitaVoci (interviene in caso di UPDATE su Voci) permettono di catturare il vincolo semantico che il campo Qta possa assumere solo valori maggiori o uguali a zero. Nel caso si cerchi di inserire un valore negativo, la query di INSERT/UPDATE fallirà perché il trigger genererà un errore in quanto cercherebbe di inserire una chiave nulla nella tabella Errori.

Document Manager: trigger 2


```
DROP TRIGGER IF EXISTS InsNumeriDdt;
CREATE TRIGGER InsNumeriDdt BEFORE
INSERT
      ON Ddt
FOR EACH ROW BEGIN
DECLARE ente CHAR(11);
DECLARE ente1 CHAR(11);
IF NEW.NColli < 0 OR NEW.PesoKg<0 OR
      (NEW.TraspAMezzo<>'cedente' AND
      NEW.TraspAMezzo<>'cessionario')
THEN
      INSERT INTO Errori(chiave) VALUES
(NULL);
END IF;
SELECT i.Ente INTO ente FROM Documenti d
      JOIN Indirizzi i ON (d.indDestinazione=i.Id)
WHERE d.Id=NEW.NumeroDoc;
SELECT Indirizzi.Ente INTO ente1 FROM
Indirizzi
      WHERE Id=NEW.Destinazione;
IF NOT (ente=ente1) THEN
INSERT INTO Errori(chiave) VALUES (NULL);
END IF;
END;
```

```
DROP TRIGGER IF EXISTS UpNumeriDdt;
CREATE TRIGGER UpNumeriDdt BEFORE
UPDATE ON Ddt
FOR EACH ROW BEGIN
DECLARE ente CHAR(11);
DECLARE ente1 CHAR(11);
IF NEW.NColli < 0 OR NEW.PesoKg<0 OR
      (NEW.TraspAMezzo<>'cedente' AND
      NEW.TraspAMezzo<>'cessionario') THEN
INSERT INTO Errori(chiave) VALUES (NULL);
END IF;
SELECT i.Ente INTO ente FROM Documenti d
JOIN
      Indirizzi i ON (d.indDestinazione=i.Id)
WHERE d.Id=NEW.NumeroDoc;
SELECT Indirizzi.Ente INTO ente1 FROM
      IndirizziWHERE Id=NEW.Destinazione;
IF NOT (ente=ente1) THEN
INSERT INTO Errori(chiave) VALUES (NULL);
END IF;
END;
```

Document Manager: trigger 2 (spiegazione)

Descrizione:

I trigger InsNumeriDdt (interviene in caso di INSERT su Ddt) e UpNumeriDdt (interviene in caso di UPDATE su Ddt) permettono di catturare i seguenti vincoli semantici:

- campo NColli >=0;
- campo PesoKg >=0;
- campo TraspAMezzo == 'cedente' OR 'cessionario';
- l'indirizzo della destinazione appartenga al destinatario del documento.
-

Nel caso si cerchi di violare uno dei precedenti vincoli, la query di INSERT/UPDATE fallirà perche' il trigger genererà un errore in quanto cercherebbe di inserire una chiave nulla nella tabella Errori.

Document Manager: trigger 3


```
DROP TRIGGER IF EXISTS InsIvaFatture;
CREATE TRIGGER InsIvaFatture BEFORE INSERT ON Fatture
FOR EACH ROW
BEGIN
      IF NEW.Iva<0
      THEN INSERT INTO Errori(chiave) VALUES (NULL);
      END IF;
END;

DROP TRIGGER IF EXISTS UpIvaFatture;
CREATE TRIGGER UpIvaFatture BEFORE UPDATE ON Fatture
FOR EACH ROW
BEGIN
      IF NEW.Iva<0
      THEN INSERT INTO Errori(chiave) VALUES (NULL);
      END IF;
END;
```

Descrizione:

- I trigger InsIvaFatture (interviene in caso di INSERT su Fatture) e UpIvaFatture (interviene in caso di UPDATE su Fatture) permettono di catturare il vincolo semantico che il campo Iva possa assumere solo valori maggiori o uguali a zero. Nel caso si cerchi di inserire un valore negativo, la query di INSERT/UPDATE fallirà perché il trigger genererà un errore in quanto cercherebbe di inserire una chiave nulla nella tabella Errori.