Andrea Tombolato, Tommaso Zagni
(1069144) (1070811)

Sommario

Abstract

La metà del cibo che viene prodotto nel mondo, circa due miliardi di tonnellate, finisce nella spazzatura benché sia in gran parte commestibile.
Fra le cause di questo spreco di massa vi sono le cattive abitudini di milioni di persone, che non conservano i prodotti in modo adeguato.
La piattaforma *eTable* nasce con lo scopo di arginare il problema tramite la gestione intelligente della dispensa personale di ogni utente.
La dispensa di un utente è composta da prodotti alimentari aventi una certa scadenza all'approssimarsi della quale sarà compito di *eTable* suggerire all'utente una lista di ricette adeguate per consumare i suddetti prodotti.
Può accadere che l'utente scelga una ricetta che utilizza anche prodotti non presenti nella sua dispensa o presenti in quantità non sufficienti, in tal caso sarà cura della piattaforma compilare una lista della spesa contenente i prodotti mancanti indicando, per ogni prodotto, l' esercizio commerciale più conveniente dove effettuare l'acquisto sulla base del prezzo applicato al prodotto d'interesse.

Descrizione Requisiti

Si vogliono realizzare un database e la relativa interfaccia web per la gestione intelligente della dispensa alimentare di un utente.

Un **tipo prodotto** è un concetto astratto che identifica una famiglia di prodotti simili.

- e.g. Una confezione di Vermicelli Pasta Zara venduta nella filiale Alì-Alìper di Piazza Metelli 6 (Padova) e una confezione di Penne rigate Barilla venduta nella filiale Coop di Via San Marco 11 (Limena) hanno entrambe tipo prodotto "Pasta".

Un **prodotto** è caratterizzato da un nome, da un prezzo di vendita al dettaglio e dalla quantità con cui si presenta, è identificato univocamente dal barcode apposto sulla confezione e dalla filiale del negozio presso cui è venduto.
Bisogna prestare attenzione al fatto che il barcode identifica un prodotto, non un esemplare di prodotto nella sua singolarità.

- e.g. il barcode 8002230000302 identifica una qualsiasi bottiglia di aperitivo Aperol da 70 cl, non una specifica bottiglia di Aperol presente in un determinato scaffale di un dato esercizio commerciale.

Ogni **negozio** è identificato dalla propria partita IVA e possiede un nome commerciale, ha inoltre almeno una **filiale** identificata dal proprio codice e dal negozio di cui è filiale.
Ogni filiale ha un indirizzo per consentirne la localizzazione.

- e.g. La partiva IVA 01515921201 identifica il negozio denominato "Coop" che ha una filiale a Limena in Via F.lli Cervi 3 con codice 0.

Si suppone che due filiali dello stesso negozio non possano avere sede nella stessa via della stessa città.
Si può pensare ad un negozio con filiali come ad una catena di negozi, mentre un negozio con un'unica filiale è qualcosa di più artigianale, come una bottega.
Filiali aventi stesso indirizzo e città sono da considerarsi come ospitate all'interno di ipermercati o strutture simili .

Ogni **utente** è identificato dal proprio username, utilizzato assieme alla password per accedere alla

piattaforma web eTable. L'utente è inoltre caratterizzato dal proprio nome e cognome.
Ciascun utente può possedere dei prodotti alimentari **in dispensa**, ognuno presente con una certa quantità ed identificato univocamente dal tipo di prodotto del quale è istanza, dall'utente che lo possiede e dalla data di scadenza.

Tramite l'interfaccia web l'utente rimane aggiornato sui propri prodotti e riceve consigli sulle ricette da utilizzare per consumare quelli in scadenza, un prodotto è considerato "**in scadenza**" se scade entro due settimane dal momento in cui si controlla.

- e.g. Se la data odierna è 01-08-2015, tutti i prodotti con data di scadenza compresa tra 01-08-2015 e 15-08-2015 sono considerati "in scadenza".

Una **ricetta** è identificata dal proprio nome ed è composta da ingredienti che sono tutti prodotti, per ogni ricetta vengono inoltre forniti i passi da seguire, una difficoltà ed un tempo indicativi per la sua preparazione.
Può accadere che un prodotto non sia ingrediente di alcuna ricetta, in tal caso verrà solo notificato all'utente l'approssimarsi della scadenza.
I piatti che è possibile preparare utilizzando le varie ricette sono divisi per **portate**: primo, secondo, contorno, dessert, spuntino.
Questa suddivisione permette di fornire all'utente una discreta scelta sulla modalità con cui consumare i prodotti che possono essere ingredienti di vari piatti.

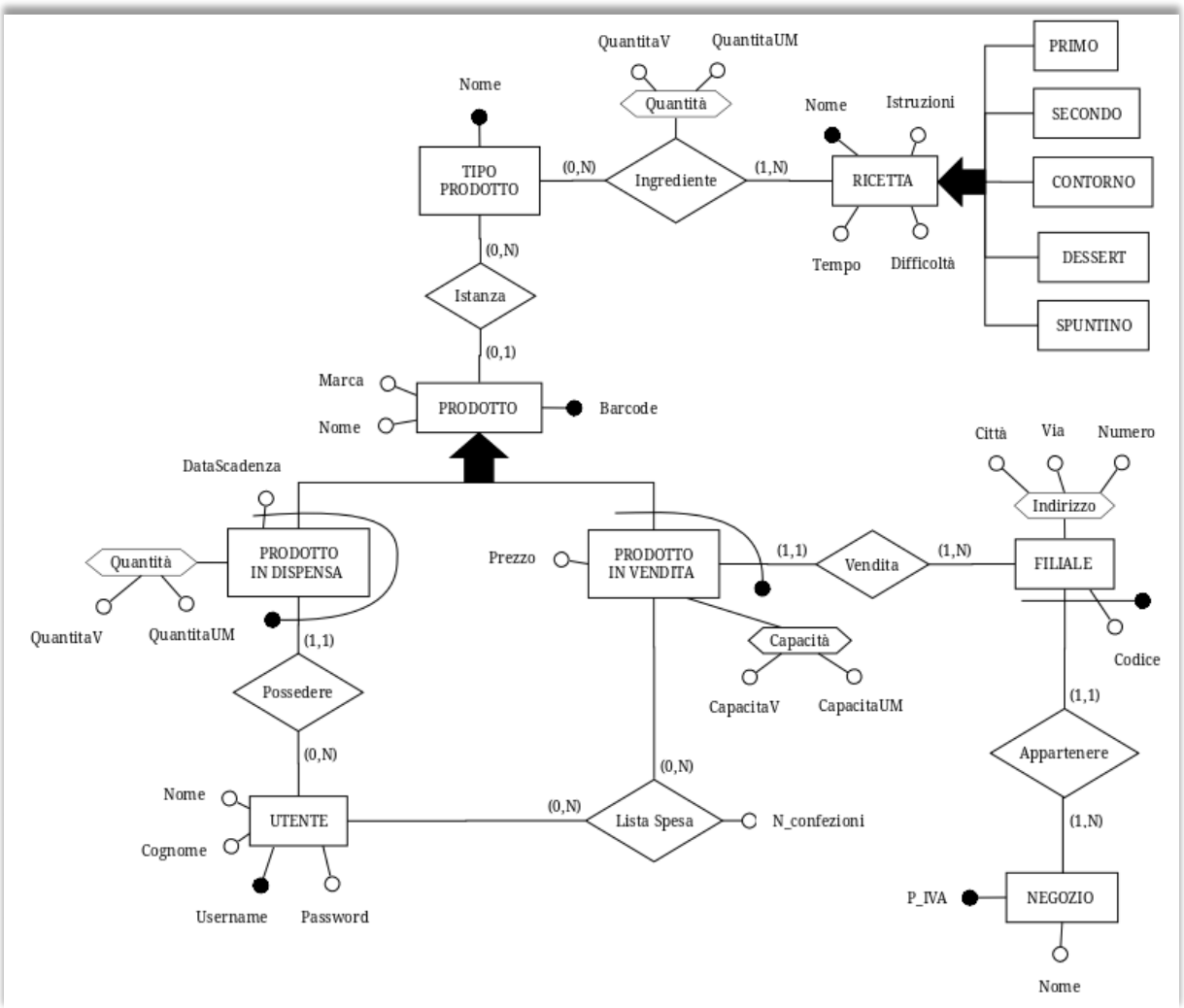
Se l'utente decide di usare una ricetta che adopera anche ingredienti non presenti nella sua dispensa verrà aggiornata la lista della spesa, aggiungendo tali prodotti mancanti.
Ogni prodotto nella **lista della spesa** è identificato dall'utente a cui è collegato, dal barcode del prodotto e dalle informazioni per raggiungere la filiale del negozio associato che propone il prezzo più conveniente.

Sarà cura di eTable stimare, per ogni prodotto nella lista della spesa, la quantità che è necessario comprare in base alla ricetta scelta e alla disponibilità in dispensa.

- e.g. Per preparare la ricetta "Carbonara al forno" per quattro persone sono necessari 350 g di pasta, la pasta conosciuta più conveniente è "Vermicelli Pasta Zara" venduta in confezioni da 500 g, l'utente non possiede alcun tipo di pasta in dispensa. In questo caso sarà sufficiente comprare un solo pacco di "Vermicelli Pasta Zara" per poter preparare la ricetta.

Progettazione Concettuale

Lo schema E-R risultante viene proposto nella figura seguente:



Dizionario dei dati

Entità	Descrizione	Attributi	Identificatore
TIPO PRODOTTO	Un tipo di prodotto.	Nome	Nome
PRODOTTO	Un prodotto reale, ovvero un prodotto fisicamente tangibile.	Barcode, Nome, Marca	Barcode
UTENTE	Usufruisce del servizio *eTable*.	Username, Password, Nome, Cognome	Username
PRODOTTO IN DISPENSA	Un Prodotto effettivamente presente nella dispensa di un determinato utente.	DataScadenza, QuantitaV, QuantitaUM	Prodotto, Utente, DataScadenza
Una *misura di quantità* è un concetto composito, formato da due atomi: QuantitaV che rappresenta il *valore numerico* della misura e QuantitaUM che rappresenta l'*unità di misura* utilizzata.			
NEGOZIO	Un tipo di negozio.	P_IVA, Nome	P_IVA
FILIALE	Istanza fisica di Negozio.	Codice, Città, Via, Numero	Codice, Negozio
PRODOTTO IN VENDITA	Un Prodotto effettivamente in vendita presso la Filiale di qualche Negozio.	Prezzo, CapacitaV, CapacitaUM	Prodotto, Filiale
Una *misura di capacità* è un concetto composito, formato da due atomi: CapacitaV che rappresenta il *valore numerico* della misura e CapacitaUM che rappresenta l'*unità di misura* utilizzata. La capacità indica "quanto" Prodotto In Vendita è contenuto nella confezione identificata da un certo Barcode ed in vendita presso una data Filiale.			
RICETTA	Fornisce istruzioni, tempo e difficoltà indicative per la preparazione di ogni ricetta.	Nome, Istruzioni, Tempo, Difficoltà	Nome

Relazione	Descrizione	Entità Coinvolte	Attributi
ISTANZA	Associa un Prodotto al suo Tipo Prodotto.	Tipo Prodotto (0,N) Prodotto (0,1)	
Un Tipo Prodotto può non essere istanziato da alcun Prodotto se, ad esempio, tale Prodotto non è momentaneamente in vendita e non è nemmeno presente nella dispensa di alcun Utente, ma lo è stato in passato. Un Prodotto può non essere istanza di alcun Tipo Prodotto se, ad esempio, un Utente inserisce un Prodotto In Dispensa che non è presente tra i Prodotti In Vendita. Questa precauzione è pensata per evitare che le informazioni sensibili del database possano essere contaminate dagli utenti inserendo dati incoerenti.			
POSSEDERE	Associa un Utente ai Prodotti che possiede.	Utente (0,N) Prodotto In Dispensa (1,1)	
VENDITA	Associa un Prodotto In Vendita alla Filiale del negozio presso cui è venduto.	Prodotto In Vendita (1,1) Filiale (1,N)	
Un Prodotto In Vendita è identificato anche dalla Filiale presso cui è venduto.			
APPARTENERE	Associa una Filiale al Negozio di cui è istanza.	Negozio (1,N) Filiale (1,1)	
LISTA SPESA	Associa un Utente ai Prodotti In Vendita dei quali è stato consigliato l'acquisto per poter preparare le ricette scelte.	Utente (0,N) Prodotto In Vendita (0,N)	N_confezioni
N_confezioni è il numero di confezioni di un certo Prodotto In Vendita che si consiglia di acquistare per poter preparare la Ricetta scelta per il numero di persone indicato. La Lista Spesa di un Utente può essere momentaneamente vuota. Un Prodotto In Vendita può non rientrare nella Lista Spesa di alcun Utente.			
INGREDIENTE	Associa una Ricetta ai Tipi Prodotto necessari per la sua preparazione.	Ricetta (1,N) Tipo Prodotto (0,N)	QuantitaV, QuantitaUM
Può capitare che non vi siano a disposizone Ricette per consumare un certo Tipo Prodotto.			

Regole di vincolo	
(RV1)	Uno specifico Prodotto In Dispensa deve appartenere ad un unico Utente.
(RV2)	Una Filiale deve vendere almeno un Prodotto.
(RV3)	Un Negozio deve avere almeno una Filiale.
(RV4)	Una Ricetta deve prevedere almeno un Ingrediente.
(RV5)	Un Prodotto in Lista Spesa deve essere un Prodotto In Vendita.
(RV6)	Una Filiale deve afferire ad un Negozio.
(RV7)	Un Prodotto In Vendita deve essere presente nella Filiale di un Negozio.
(RV8)	Un Prodotto In Dispensa deve avere una Data di Scadenza
(RV9)	Un Ingrediente deve afferire ad una Ricetta.
(RV10)	La QuantitaV di un Prodotto In Dispensa deve essere maggiore di zero.

Regole di derivazione	
(RD1)	Data *Q* quantità di un certo Prodotto da comprare per poter preparare una specifica Ricetta e *C* capacità della confezione contenente tale Prodotto In Vendita, allora N_confezioni di un Prodotto in Lista Spesa si ottiene come *Q/C*.

Progettazione Logica

Ristrutturazione dello schema E-R

Sui dati descritti dallo schema E-R sono previste le seguenti operazioni (tramite interfaccia web):

Operazioni	
(OP1)	Inserire un nuovo Utente indicando tutti i suoi dati.
(OP2)	Trovare i Prodotti In Dispensa per un determinato Utente.
(OP3)	Trovare i Prodotti in scadenza per un determinato Utente.
(OP4)	Per un determinato Utente, trovare le Ricette che hanno, tra i loro Ingredienti, almeno un Prodotto in scadenza appartenente a tale Utente.
(OP5)	Inserire un nuovo Prodotto In Dispensa.
(OP6)	Eliminare un Prodotto In Dispensa.
(OP7)	Trovare i dettagli relativi ad una certa Ricetta.
(OP8)	Dato un Prodotto In Dispensa presente con quantità Q_0 ed un numero reale positivo q, modificare la quantità del Prodotto In Dispensa in $Q_1=Q-q$.
(OP9)	Data una Ricetta ed un numero intero n, inserire in Lista Spesa il Prodotto In Vendita più conveniente e la relativa quantità da comprare per poter preparare tale Ricetta.
(OP10)	Eliminare un prodotto da Lista Spesa.

Analisi delle ridondanze

Nello schema E-R originale non sono presenti ridondanze.

Eliminazione delle gerarchie

Dato che i sistemi tradizionali per la gestione di basi di dati non consentono la rappresentazione diretta di una gerarchia, si deve trasformare tale costrutto in modo che possa essere implementato.
Nello schema sono presenti due gerarchie: quella relativa alle Ricette e quella relativa ai Prodotti.
Per le Ricette si può notare che le operazioni che le riguardano non fanno distinzioni tra le varie portate, tra l'altro le entità corrispondenti non hanno attributi specifici che le distinguono.
Conviene quindi accorpare le entità figlie della generalizzazione nel genitore aggiungendo l'attributo "Portata" all'entità Ricetta che ha un dominio costituito dai nomi "Primo", "Secondo", "Contorno", "Dessert", "Spuntino".

Per i Prodotti conviene accorpare l'entità genitore della generalizzazione nelle figlie, eliminando quindi Prodotto e facendo ereditare i suoi attributi e le relazioni a cui partecipava a Prodotto in Dispensa e Prodotto in Vendita. Questa scelta conviene, rispetto a quella di sostituire la generalizzazione con delle associazioni, perchè evita di dover accedere sia a Prodotto che a Prodotto che ad una della sue figlie ogniqualvolta abbia bisogno degli attributi Marca, Nome, Barcode.

Partizionamento di concetti

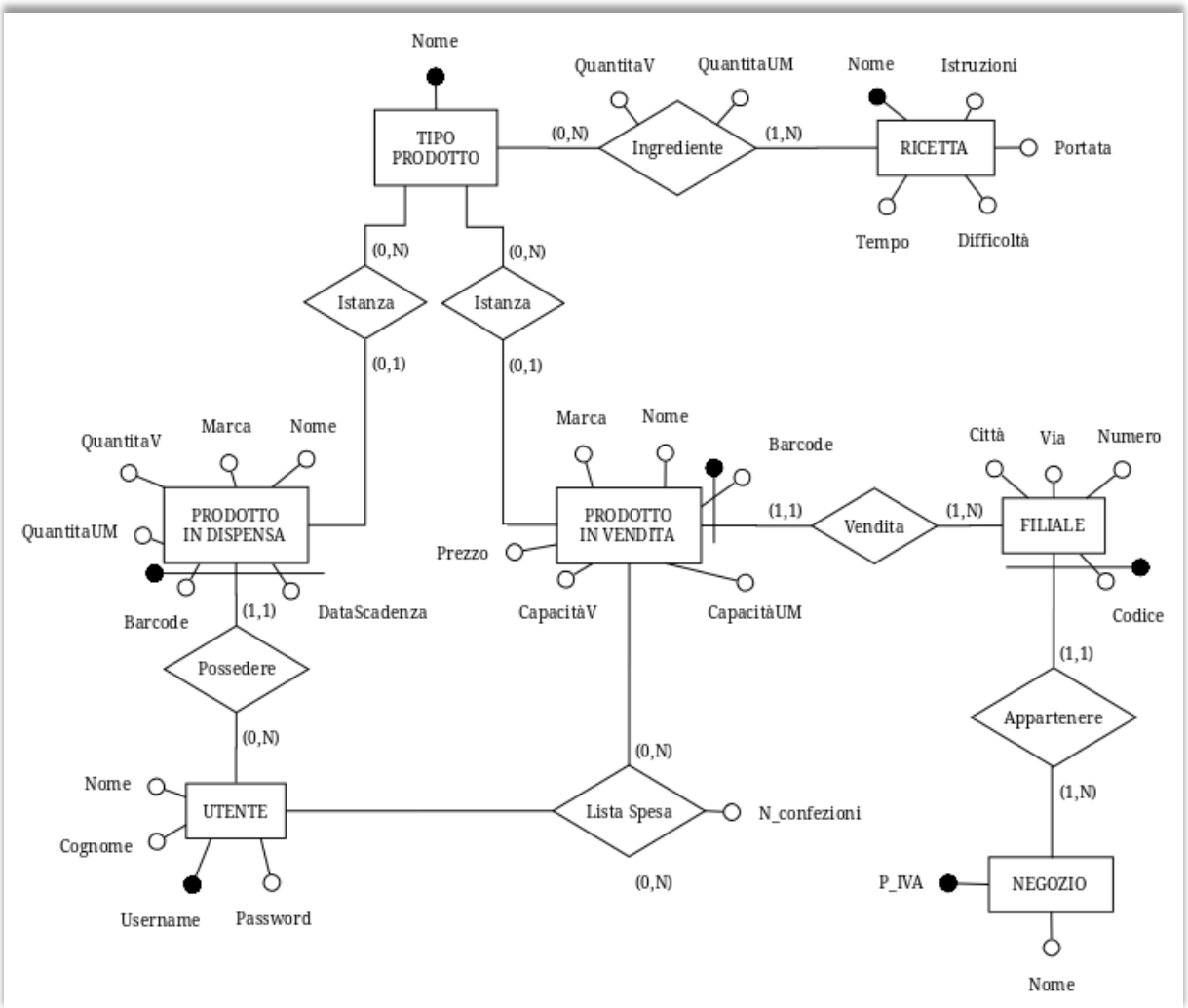
Dato che i sistemi tradizionali per la gestione di basi di dati non consentono la rappresentazione diretta di un attributo composto, si deve trasformare tale costrutto in modo che possa essere implementato.
Nello schema sono presenti quattro attributi composti che vengono rimossi e sostituiti dai singoli attributi che raggruppano: Capacità di Prodotto In Vendita viene sostituito da CapacitaV e CapacitaUM, Quantità di Prodotto In Dispensa viene sostituito da QuantitaV e QuantitaUM, Quantità di Ingrediente viene sostituito da QuantitaV e QuantitaUM, Indirizzo di Filiale viene sostiutito da Città, Via Numero.

Scelta degli identificatori principali

Non ci sono ambiguità nella scelta degli identificatori in quanto ogni entità ne presenta solo uno, con il significato spiegato di seguito.

Entità	Identificatore	Significato
TIPO PRODOTTO	Nome	Ogni Tipo Prodotto è identificato univocamente dal proprio Nome.
UTENTE	Username	Ogni Utente è identificato univocamente dal proprio Username.
PRODOTTO IN DISPENSA	Barcode, Utente, DataScadenza	Barcode e DataScadenza sono attributi interni, mentre Utente indica che tale entità usa anche l'identificatore di Utente come attributo esterno per creare il proprio identificatore principale. In questo modo Utenti diversi possono avere in dispensa prodotti simili, inoltre un Utente può mantenere in dispensa prodotti simili ma con date di scadenza diverse.
PRODOTTO IN VENDITA	Barcode, Filiale	Barcode è attributo interno mentre Filiale indica che tale entità usa anche l'identificatore di Filiale come attributo esterno per creare il proprio identificatore principale. In questo modo può essere rappresentato il caso di prodotti simili venduti presso Filiali diverse.
NEGOZIO	P_IVA	Ogni Negozio è identificato univocamente dalla propria P_IVA.
FILIALE	Codice, Negozio	Codice è attributo interno mentre Negozio indica che tale entità usa anche l'identificatore di Negozio come attributo esterno per creare il proprio identificatore principale.
RICETTA	Nome	Ogni Ricetta è identificata univocamente dal proprio Nome.

Abbiamo con questo terminato la fase di ristrutturazione dello schema E-R originale, ne risulta lo schema seguente:



Traduzione dello schema E-R

Lo schema E-R ristrutturato può essere tradotto nel seguente schema relazionale:

Utente (Username, Pasword, Nome, Cognome)

TipoProdotto (Nome)

Ricetta (Nome, Portata, Tempo, Difficoltà, Istruzioni)

Negozio (P_IVA, Nome)

Filiale (Codice, Negozio, Città, Via, Numero)

ProdottoInVendita (Barcode, Filiale, Negozio, TipoProdotto, Prezzo, Marca, Nome, CapacitaV, CapacitaUM)

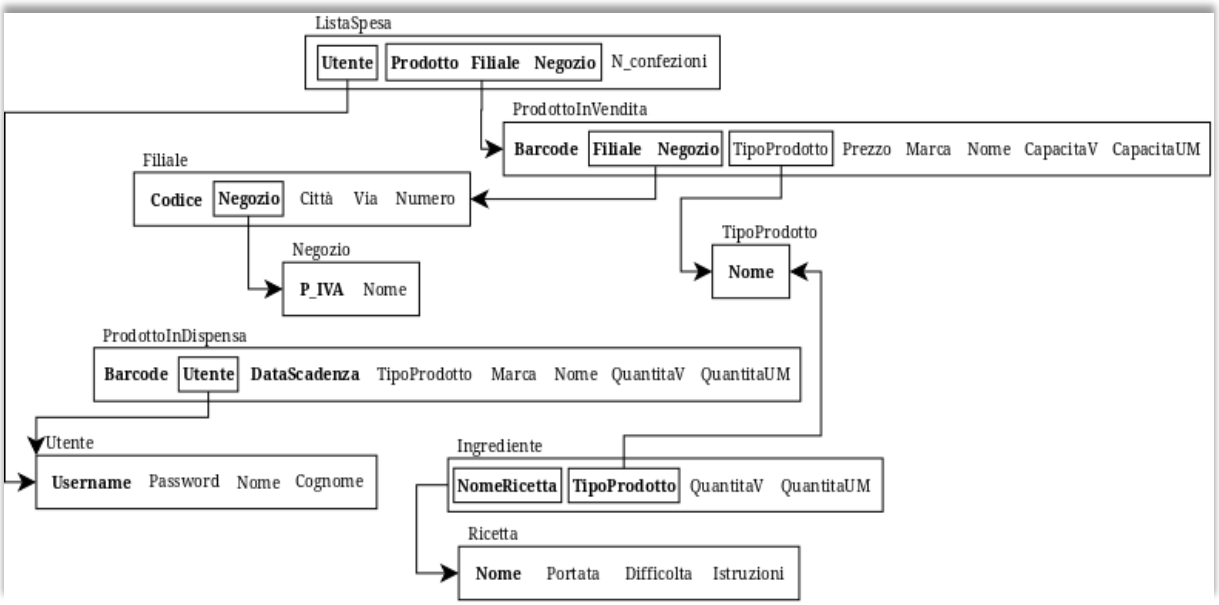
ProdottoInDispensa (Barcode, Utente, DataScadenza, TipoProdotto, Marca, QuantitaV, QuantitaUM)

Ingrediente (NomeRicetta, TipoProdotto ,QuantitaV, QuantitaUM)

ListaSpesa (Utente, Prodotto, Filiale, Negozio, N_confezioni)

Descrizione dei vincoli di integrità referenziale

Nel processo di traduzione dello schema E-R originale a schema logico sono stati introdotti dei vincoli di integrità referenziale il cui significato è spiegato di seguito.



In questo diagramma le chiavi delle relazioni sono rappresentate in grassetto mentre le frecce indicano vincoli di integrità referenziale.

Implementazione

Creazione delle tabelle

```
TABLE Utente(
    Username VARCHAR(20)  PRIMARY KEY,
    Password VARCHAR(20)  NOT NULL,
    Nome VARCHAR(20)  NOT NULL,
    Cognome VARCHAR(20)  NOT NULL
)
ENGINE = INNODB;

CREATE TABLE TipoProdotto(
    Nome VARCHAR(50) PRIMARY KEY
)
ENGINE = INNODB;

CREATE TABLE Ricetta(
    Nome VARCHAR(100)  PRIMARY KEY,
    Portata ENUM('Primo','Secondo','Spuntino','Dessert','Contorno') NOT NULL,
    Tempo TIME  NOT NULL,
    Difficolta ENUM('Bassa','Media','Alta') NOT NULL,
    Istruzioni TEXT
)
ENGINE = INNODB;
```

Viene data la possibilità di lasciare a NULL il valore dell'attributo Istruzioni in modo da poter inserire il TEXT ad esso corrispondente in un secondo momento, dopo aver creato la *entry* di Ricetta nel database.
Per l'attributo Istruzioni è stato scelto il tipo TEXT in quanto può contenere una stringa di 65.535 caratteri, abbastanza per descrivere i passi da seguire per preparare una Ricetta.

```
CREATE TABLE Negozio(
    P_IVA CHAR(11) PRIMARY KEY,
    Nome VARCHAR(50)  NOT NULL
)
ENGINE = INNODB;
```

L'attributo P_IVA di Negozio ha tipo CHAR(11) perchè si è scelto di rappresentare la sola realtà italiana, nella quale l'identificativo di partita IVA si compone di 11 cifre.

```
CREATE TABLE Filiale(
    Codice SMALLINT UNSIGNED,
    Negozio CHAR(11)  NOT NULL,
    Città VARCHAR(20)  NOT NULL,
    Via VARCHAR(50)  NOT NULL,
    Numero SMALLINT  NOT NULL,

    FOREIGN KEY (Negozio) REFERENCES Negozio(P_IVA) ON DELETE CASCADE ON UPDATE CASCADE,

    PRIMARY KEY(Codice,Negozio)
)
ENGINE = INNODB;
```

```
CREATE TABLE ProdottoInVendita(

    Barcode CHAR(13),
    Filiale SMALLINT UNSIGNED,
    Negozio CHAR(11),
    TipoProdotto VARCHAR(50),
    Prezzo DECIMAL(7,2) NOT NULL,
    Marca VARCHAR(20) NOT NULL,
    Nome VARCHAR(50) NOT NULL,
    CapacitaV FLOAT NOT NULL,
    CapacitaUM VARCHAR(2) NOT NULL,

    FOREIGN KEY (Filiale,Negozio)
            REFERENCES Filiale(Codice,Negozio) ON DELETE CASCADE ON UPDATE CASCADE,

    FOREIGN KEY (TipoProdotto)
            REFERENCES TipoProdotto(Nome) ON DELETE CASCADE ON UPDATE CASCADE,

    PRIMARY KEY(Barcode,Filiale,Negozio)
)
ENGINE = INNODB;
```

L'attributo CapacitaUM di Negozio (similmente QuantitaUM di ProdottoInDispensa) ha tipo VARCHAR(2) per permettere di inserire i simboli del SI (“Sistema Internazionale ”) corrispondenti alle varie unità di misura utilizzate per descrivere la quantità di un certo prodotto: g, kg, l, ml.

```
CREATE TABLE ProdottoInDispensa(
    Barcode CHAR(13),
    Utente VARCHAR(20),
    DataScadenza DATE NOT NULL,
    TipoProdotto VARCHAR(50) NOT NULL,
    Marca VARCHAR(20) NOT NULL,
    Nome VARCHAR(50) NOT NULL,
    QuantitaV FLOAT NOT NULL,
    QuantitaUM VARCHAR(2) NOT NULL,

    FOREIGN KEY (Utente) REFERENCES Utente(Username) ON DELETE CASCADE ON UPDATE CASCADE,

    PRIMARY KEY(Barcode,Utente,DataScadenza)
)
ENGINE = INNODB;
```

Triggers

converti_UM

Nella tabella ProdottoInDispensa si è deciso di utilizzare, come unità di misura fondamentali, rispettivamente i grammi (g) per i prodotti solidi e i millilitri (ml) per i prodotti liquidi.
Prima di ogni inserimento nella suddetta tabella è quindi necessario che un trigger controlli che l'unità di misura relativa al prodotto che si sta inserendo sia *g* o *ml*, se invece è *kg* o *l* il trigger si occupa di aggiornare l'unita di misura convertendone il relativo valore verso *g* o *ml*.
È stato scelto di gestire solo le conversioni da *kg* e *l* perchè reputate le unità di misura più frequenti per descrivere quantità relative a prodotti alimentari, il trigger è comunque estendibile con ulteriori unità di misura.
Se si tenta di inserire unità di misura diverse da quelle supportate o da *' '* (*carattere vuoto*, utilizzato per prodotti dove l'unità di misura non è importante, ad esempio le uova) viene restituito un messaggio d'errore.

```
CREATE TRIGGER `converti_UM`
BEFORE INSERT ON `ProdottoInDispensa`
FOR EACH ROW
BEGIN
        IF new.QuantitaUM = 'kg' THEN
                SET new.QuantitaUM = 'g';
            SET new.QuantitaV = new.QuantitaV * 1000;
    ELSE
        IF new.QuantitaUM = 'l' THEN
        SET new.QuantitaUM = 'ml';
        SET new.QuantitaV = new.QuantitaV * 1000;
        ELSE
        IF (new.QuantitaUM <> 'ml' AND new.QuantitaUM <> 'g'
                            AND new.QuantitaUM <> ' ') THEN

                        INSERT INTO ProdottoInDispensa (SELECT * FROM ProdottoInDispensa);
            END IF;
        END IF;
    END IF;
END
```

inserisci_nuovo_tipo

Esiste un vincolo di integrità referenziale tra l'attributo TipoProdotto di ProdottoInVendita e l'entità TipoProdotto, per agevolare l'inserimento in ProdottoInVendita si è pensato a questo trigger che, se il TipoProdotto del ProdottoInVendita che si sta andando ad inserire non è già presente nella tabella TipoProdotto, prima di inserire la entry in ProdottoInVendita aggiorna la tabella TipoProdotto con il relativo tipo mancante.

```
CREATE DEFINER TRIGGER `inserisci_nuovo_tipo`
BEFORE INSERT ON `ProdottoInVendita`
FOR EACH ROW
BEGIN
    IF NOT EXISTS (SELECT * FROM TipoProdotto WHERE Nome = New.TipoProdotto) THEN

        INSERT INTO TipoProdotto VALUES (New.TipoProdotto);

    END IF;
END
```

Procedure

aggiusta_dispensa

sia *prodotto* un ProdottoInVendita con una certa DataScadenza *scadenza*, *username* un Utente e *confezioni* un certo numero di confezioni delle quali è stato consigliato l'acquisto, questa procedura controlla se *prodotto* è già presente nella dispensa di *username* con la medesima *scadenza* oppure no.
Nel primo caso viene eseguito un UPDATE della riga individuata facendo:

$$QuantitaV = QuantitaV + CapacitaV * confezioni$$

con CapacitaV relativa a *prodotto*.
Nel secondo caso viene eseguito un INSERT del nuovo *prodotto* usando gli argomenti passati come paramentro alla procedura.
Nella pratica, questa procedura serve ad evitare la generazione di errori dovuti al tentato inserimento di entry duplicate.
Questa procedura è richiamata per ogni inserimento in ProdottiInDispensa.

```
CREATE PROCEDURE `aggiusta_dispensa`( `prodotto` CHAR(13),  `scadenza` DATE, `username` VARCHAR(20),
`confezioni` INT)
BEGIN

  DECLARE var_tipo_prodotto VARCHAR(50);
  DECLARE var_marca VARCHAR(20);
  DECLARE var_nome VARCHAR(50);
  DECLARE var_quantita_V FLOAT UNSIGNED;
  DECLARE var_quantita_UM CHAR(2);
  DECLARE quant FLOAT(5) UNSIGNED;

SELECT CapacitaV * confezioni
  INTO quant
  FROM ProdottoInVendita
  WHERE Barcode = prodotto
  LIMIT 1;

  IF EXISTS (SELECT *
               FROM ProdottoInDispensa P
            WHERE P.Utente = username AND P.Barcode = prodotto
                AND P.DataScadenza = scadenza) THEN

            UPDATE ProdottoInDispensa
            SET QuantitaV = QuantitaV + quant
            WHERE Utente = username AND Barcode = prodotto AND DataScadenza = scadenza;

  ELSE/*In ProdottoInDispensa non c'è riga con stesso Utente-Barcode-Scadenza*/

            SELECT DISTINCT P.TipoProdotto,P.Marca,P.Nome,P.CapacitaUM
            INTO var_tipo_prodotto,var_marca,var_nome,var_quantita_UM
            FROM ProdottoInVendita P
            WHERE P.Barcode = prodotto;

            INSERT INTO ProdottoInDispensa(Barcode,Utente,DataScadenza,TipoProdotto,
                                           Marca,Nome, QuantitaV,QuantitaUM)
                    VALUES(prodotto,username,scadenza,var_tipo_prodotto,var_marca,var_nome,
                           quant,var_quantita_UM);

  END IF;
END
```

acquista_prodotto

sia *barcode* un prodotto nella ListaSpesa dell'Utente *username* con una certa DataScadenza *scadenza,* questa procedura cancella da ListaSpesa la riga relativa a *barcode* ed *username* andandola ad inserire in ProdottoInDispensa avvalensosi della procedura *aggiusta_dispensa*.
Nella pratica, questa procedura simula il passaggio di un ProdottoInVendita dal bancone del Negozio alla dispensa personale di un Utente.
Questa procedura è richiamata ogni volta che Utente decide di acquistare un prodotto che gli è stato suggerito attraverso ListaSpesa.

```
CREATE PROCEDURE `acquista_prodotto`( `username` VARCHAR(20), `barcode` CHAR(13),`scadenza` DATE)
BEGIN

  DECLARE var_tipo_prodotto VARCHAR(50);
  DECLARE var_marca VARCHAR(20);
  DECLARE var_nome VARCHAR(50);
  DECLARE var_quantita_V FLOAT UNSIGNED;
  DECLARE var_quantita_UM CHAR(2);
  DECLARE var_confezioni SMALLINT;

  SELECT L.N_confezioni
  INTO var_confezioni
  FROM ListaSpesa L
  WHERE L.Prodotto = barcode;

      DELETE FROM ListaSpesa
            WHERE Utente = username AND Prodotto = barcode;

      CALL aggiusta_dispensa(barcode,scadenza,username,var_confezioni);

END
```

aggiusta_lista_spesa

similmente al comportamento di *aggiusta_dispensa* per ProdottoInDispensa, questa procedura si occupa di gestire l'aggiunta dei prodotti in ListaSpesa: se l'articolo *prodotto* è già presente nella ListaSpesa dell'Utente *username* allora si incrementerà la relativa quantità delle confezioni da acquistare, altrimenti verrà inserita una nuova riga in ListaSpesa di *username* riguardante l'articolo *prodotto*.

```
CREATE PROCEDURE `aggiusta_lista_spesa`(`username` VARCHAR(20),`prodotto` CHAR(13),  `confezioni` SMALLINT)
BEGIN
            DECLARE var_prezzo DECIMAL(7,2);
            DECLARE var_filiale SMALLINT;
            DECLARE var_negozio CHAR(11);

     IF EXISTS( SELECT *
               FROM ListaSpesa L
               WHERE L.Utente = username AND L.Prodotto = prodotto) THEN

       UPDATE ListaSpesa
            SET N_confezioni = N_confezioni + confezioni
            WHERE Utente = username AND Prodotto = prodotto;
  ELSE
            SELECT MigliorPrezzo,Filiale,Negozio
            INTO var_prezzo,var_filiale,var_negozio
            FROM ConsigliAcquisti C
            WHERE C.Barcode = prodotto;

            INSERT INTO ListaSpesa(Utente,Prodotto,Filiale,Negozio,N_confezioni)
                  VALUES(username,prodotto,var_filiale,var_negozio,confezioni);
  END IF;
END
```

elimina_prodotto_finito

Un ProdottoInDispensa è considerato *"finito"* se ha QuantitaV non positiva, questa procedura si occupa di eliminare da ProdottoInDispensa i prodotti finiti appartenenti all'Utente *username*.

```
CREATE PROCEDURE `elimina_prodotto_finito`(`username` VARCHAR(20))
BEGIN

 WHILE EXISTS (SELECT * FROM ProdottoInDispensa WHERE QuantitaV <= 0 and Utente = username) DO
  DELETE FROM ProdottoInDispensa WHERE QuantitaV <= 0 and Utente = username;
 END WHILE;

END
```

Funzioni

autocomplete_prod_info

Sia il prodotto *brcode* con DataScadenza *scadenza*, questa funzione controlla se in ProdottoInVendita esiste il prodotto *brcode* oppure no.
Nel primo caso, *brcode* viene inserito in ProdottoInDispensa usando il relativo ProdottoInVendita per completare le informazioni mancanti (TipoProdotto, Marca, Nome, CapacitaV, CapacitaUM) e viene restituito TRUE, nel secondo caso viene semplicemente restituito FALSE.
Questa funzione è stata pensata per agevolare e velocizzare l'inserimento in ProdottoInDispensa da parte dell'Utente, in modo che Utente debba inserire manualmente tutte le informazioni relative a *barcode* se e solo se il prodotto *barcode* non è già conosciuto dal database.

```
CREATE FUNCTION `autocomplete_prod_info`(`brcode` CHAR(13), `usr` VARCHAR(20), `scadenza` DATE)
RETURNS BOOLEAN
BEGIN

 DECLARE var_tipo_prodotto VARCHAR(50) DEFAULT '-1';
 DECLARE var_marca VARCHAR(20);
 DECLARE var_nome VARCHAR(50);
 DECLARE var_quantita_V FLOAT UNSIGNED;
 DECLARE var_quantita_UM CHAR(2);

 SELECT DISTINCT P.TipoProdotto,P.Marca,P.Nome,P.CapacitaV,P.CapacitaUM
 INTO var_tipo_prodotto,var_marca,var_nome,var_quantita_V,var_quantita_UM
 FROM ProdottoInVendita P
 WHERE P.Barcode = brcode;

 IF (var_tipo_prodotto = '-1') THEN
  RETURN false;
 ELSE
  CALL aggiusta_dispensa(brcode,scadenza,usr,'1');
  RETURN true;
 END IF;

END
```

presente_in_dispensa

Questa funzione ritorna TRUE se e solo se l'articolo *prodotto* con DataScadenza *scadenza* è presente nella dispensa dell'Utente *username*, altrimenti ritorna FALSE.

```
CREATE FUNCTION `presente_in_dispensa`(`prodotto` CHAR(13), `username` VARCHAR(20), `scadenza`
                                       DATE)

RETURNS BOOLEAN
BEGIN
    IF EXISTS (SELECT * FROM ProdottoInDispensa P  WHERE P.Utente = username AND P.Barcode = prodotto
            AND P.DataScadenza = scadenza) THEN

        RETURN TRUE;
        ELSE

                RETURN FALSE;

         END IF;

END
```

is_recipe_available

Questa funzione restituisce TRUE se e solo se nella dispensa dell'Utente *usr* sono presenti tutti gli Ingredienti (e nelle QuantitaV necessarie) per poter preparare la Ricetta *ricetta* per *n_pers* persone, altrimenti restituisce FALSE.

```
CREATE FUNCTION `is_recipe_available`(`ricetta` VARCHAR(100), `n_pers` INT UNSIGNED,
                                      `usr` VARCHAR(20))

RETURNS BOOLEAN
BEGIN
    IF EXISTS(SELECT TipoProdotto
              FROM Ingrediente
              WHERE NomeRicetta = ricetta
                    AND TipoProdotto NOT IN (SELECT TipoProdotto
                                             FROM ProdottoInDispensa
                                             WHERE Utente = usr)) THEN
        RETURN FALSE;
    ELSE
        IF EXISTS(SELECT*
                  FROM (SELECT Barcode, PD.TipoProdotto,SUM(PD.QuantitaV) AS QuantitaDispensa,
                               I.QuantitaV AS QuantitaNecessaria,
                               SUM(PD.QuantitaV)-(I.QuantitaV/4) * n_pers AS  QuantitaRimanente
                        FROM Ingrediente I JOIN ProdottoInDispensa PD ON I.TipoProdotto = PD.TipoProdotto
                        WHERE I.NomeRicetta = ricetta AND PD.Utente = usr
                        GROUP BY PD.TipoProdotto) AS T
                  WHERE T.QuantitaRimanente < 0) THEN

                    RETURN FALSE;

        ELSE

                    RETURN TRUE;

            END IF;
        END IF;
END
```

L'interrogazione:

```
EXISTS(SELECT TipoProdotto
            FROM Ingrediente
            WHERE NomeRicetta = ricetta
                  AND TipoProdotto NOT  IN (SELECT TipoProdotto
                                            FROM ProdottoInDispensa
                                            WHERE Utente = usr))
```

restituisce TRUE se e solo se c'è almeno un TipoProdotto di Ingrediente utilizzato per *ricetta* che manca tra i TipoProdotto dei prodotti in dispensa dell'Utente *usr*.
L'interrogazione:

```
EXISTS(SELECT*
                  FROM (SELECT Barcode, PD.TipoProdotto,SUM(PD.QuantitaV) AS QuantitaDispensa,
                              I.QuantitaV AS QuantitaNecessaria,
                              SUM(PD.QuantitaV)-(I.QuantitaV/4) * n_pers AS  QuantitaRimanente
                        FROM Ingrediente I JOIN ProdottoInDispensa PD ON I.TipoProdotto = PD.TipoProdotto
                        WHERE I.NomeRicetta = ricetta AND PD.Utente = usr
                        GROUP BY PD.TipoProdotto) AS T
                  WHERE T.QuantitaRimanente < 0)
```

restituisce TRUE se e solo se c'è almeno un TipoProdotto di Ingrediente utilizzato per *ricetta* la cui QuantitaV non è sufficiente a preparare *ricetta* per *n_pers* persone.

Interrogazioni

Selezionare i prodotti presenti nella dispensa dell'Utente '*mconti*' , concatenando QuantitaV e QuantitaUM sotto l'unico denominatore “Quantità” e visualizzando DataScadenza nel formato “31-07-2015”.
I risultati vengono mostrati avendo cura di ordinarli per DataScadenza crescente in modo da dare risalto ai prodotti che scadono prima.

```
SELECT Barcode,Marca,Nome,TipoProdotto AS Tipo,
    CONCAT(QuantitaV,' ',QuantitaUM) AS Quantità,
    DATE_FORMAT(DataScadenza,'%e-%c-%Y') AS Scadenza
FROM ProdottoInDispensa
WHERE Utente='mconti'
ORDER BY DataScadenza ASC
```

Selezionare, tra i prodotti nella dispensa dell'Utente '*mconti*' , quelli “in scadenza” , concatenando QuantitaV e QuantitaUM sotto l'unico denominatore “Quantità” e visualizzando DataScadenza nel formato
“31-07-2015”.
I risultati vengono mostrati avendo cura di ordinarli per DataScadenza crescente, risultati con DataScadenza uguale vengono ordinati per QuantitaV decrescente in modo da dare risalto ai prodotti che scadono prima e che sono presenti in quantità più massicce in dispensa.

```
SELECT Marca,Nome,TipoProdotto AS Tipo,CONCAT(QuantitaV,' ',QuantitaUM) AS Quantità,
    DATE_FORMAT(DataScadenza,'%e-%c-%Y') AS Scadenza
FROM ProdottoInDispensa
WHERE Utente = 'mconti' AND DataScadenza < DATE_ADD(CURDATE(),INTERVAL 2 WEEK)
ORDER BY DataScadenza,QuantitaV DESC
```

Selezionare le Ricette che annoverano tra i loro Ingredienti TipiProdotto compatibili con quelli dei ProdottiInDispensa dell'Utente '*mconti*' che stanno scadendo, mostrando Nome,Portata,Tempo,Difficoltà della Ricetta ed il numero di TipiProdotto diversi consumati dalla relativa Ricetta rapportato al totale dei diversi TipiProdotto in scadenza che, per questo esempio, è stato supposto essere 4.
I risultati vengono mostrati avendo cura di ordinarli per numero di ProdottiConsumati decrescente.
Nella pratica, questa interrogazione seleziona le Ricette adatte a consumare i prodotti in scadenza dell'Utente '*mconti*', mostrando le informazioni interessanti per ogni ricetta ed ordinando i risultati in modo da dare rilevanza alle Ricette che utilizzano un insieme più ampio di TipiProdotto in scadenza.

```
SELECT R.Nome,R.Portata,R.Tempo,R.Difficolta AS Difficoltà,
    CONCAT(COUNT(DISTINCT PD.TipoProdotto),' su ', 4) AS ProdottiConsumati
FROM (Ingrediente I JOIN Ricetta R ON I.NomeRicetta = R.Nome)
        JOIN ProdottoInDispensa PD ON I.TipoProdotto = PD.TipoProdotto
WHERE PD.Utente = 'mconti' AND PD.DataScadenza < DATE_ADD(CURDATE(),INTERVAL 2 WEEK)
GROUP BY R.Nome
ORDER BY COUNT(*) DESC
```

Selezionare tutti i prodotti nella ListaSpesa dell'utente '*mconti*' mostrando : Barcode, Nome e Marca (concatenati in Prodotto), il Nome del Negozio che vende il prodotto, l'Indirizzo della Filiale (composto da Via, Numero, Città), il Prezzo di vendita in quella Filiale e il numero di confezioni che si consiglia di acquistare.

```
SELECT DISTINCT PV.Barcode, CONCAT(PV.Nome,' ',PV.Marca) AS Prodotto,N.Nome AS Negozio,
        CONCAT(F.Via,' ',F.Numero,' (',F.Città,')') AS Indirizzo, PV.Prezzo,L.N_confezioni
FROM ((ListaSpesa L JOIN ProdottoInVendita PV ON L.Prodotto = PV.Barcode)
          JOIN Negozio N ON L.Negozio = N.P_IVA) JOIN Filiale F ON L.Filiale = F.Codice
WHERE F.Negozio = N.P_IVA AND L.Utente = 'mconti' AND PV.Prezzo = ( SELECT MIN(PV.Prezzo)
                                                                    FROM ProdottoInVendita PV
                                                                    WHERE PV.Barcode = L.Prodotto)
GROUP BY PV.Barcode
```

Creare una vista che contenga i TipiProdotto mancanti nella dispensa dell'Utente '*mconti*' e necessari alla preparazione nella Ricetta 'Carbonara al forno' per 3 persone indicando, per ogni TipoProdotto, il Nome, la quantità per 4 persone e la quantità effettivamente mancante ottenuta come:

*QuantitaMancante = (QuantitaPer4Pers/4) * 3 - SUM(QuantitaDispensa)*

Per creare tale vista si unisce il risultato di due interrogazioni: la prima seleziona i TipoProdotto totalmente mancanti nella dispensa di '*mconti*' e necessari alla preparazione nella Ricetta 'Carbonara al forno' per 3 persone, mentre la seconda restituisce le quantità mancati per i TipoProdotto presenti in dispensa ma con quantità insufficienti.

```
CREATE VIEW TipiProdottoMancanti AS
 SELECT TipoProdotto,(QuantitaV/4) * 3 AS QuantitaMancante, QuantitaV AS QuantitaPer4Pers
 FROM Ingrediente
 WHERE NomeRicetta = 'Carbonara al forno'
          AND TipoProdotto NOT IN (SELECT TipoProdotto
                                   FROM ProdottoInDispensa
                                   WHERE Utente = 'mconti')
 UNION

 SELECT PD.TipoProdotto, ((I.QuantitaV/4) * 3-SUM(PD.QuantitaV)) AS QuantitaMancante,
     I.QuantitaV AS QuantitaPer4Pers
 FROM Ingrediente I JOIN ProdottoInDispensa PD ON I.TipoProdotto = PD.TipoProdotto
 WHERE I.NomeRicetta = 'Carbonara al forno' AND PD.Utente = 'mconti'
 GROUP BY PD.TipoProdotto
 HAVING SUM(PD.QuantitaV) < (I.QuantitaV/4) * 3
```

Creare una vista che contenga , per ogni TipoProdotto della vista TipiProdottiMancanti, il Nome del TipoProdotto, la sua quantità mancante, la CapacitaV del relativo ProdottoInVendita, il prezzo conosciuto più conveniente, ottenuto dalla ricerca in ProdottoInVendita.
Il "prodotto più conveniente" non è semplicemente quello con il prezzo minore per un certo TipoProdotto, bensì il ProdottoInVendita che presenta il minor rapporto:

Prezzo/CapacitaV

```
CREATE VIEW MigliorPrezzoPerTipo AS
  SELECT T.TipoProdotto, MIN(Prezzo) AS MigliorPrezzo, QuantitaMancante,PV.CapacitaV
  FROM TipiProdottoMancanti T JOIN ProdottoInVendita PV ON T.TipoProdotto = PV.TipoProdotto
  WHERE (Prezzo/PV.CapacitaV) = ( SELECT MIN(Prezzo/P.CapacitaV)
                                  FROM TipiProdottoMancanti TPM JOIN ProdottoInVendita P
                                       ON TPM.TipoProdotto = P.TipoProdotto
                                  WHERE P.TipoProdotto = PV.TipoProdotto)
 GROUP BY PD.TipoProdotto
```

Creare una vista che contenga , per ogni TipoProdotto della vista MigliorPrezzoPerTipo, tutte le informazioni essenziali sui ProdottiInVendita dei quali viene consigliato l'acquisto.
Si è scelto di arrotondare per eccesso tramite la funzione CEIL qualsiasi valore decimale risultante dal rapporto:

M.QuantitaMancante/PV.Capacità

in modo da non rischiare che le ConfezioniDaComprare consigliate siano insufficienti rispetto alla reale necessità.
Ad esempio, nella situazione:

M.QuantitaMancante/PV.Capacità = 1,1

se non si utilizzasse CEIL, ConfezioniDaComprare risulterebbe uguale ad 1, rovinando irrimediabilmente l'esperienza utente che si ritroverebbe ad aver acquistato gli ingredienti con quantità insufficiente a preparare la ricetta scelta.

```
CREATE VIEW ConsigliAcquisti AS
  SELECT MigliorPrezzo, Barcode, Filiale, Negozio,
         CEIL(M.QuantitaMancante/PV.CapacitaV) AS ConfezioniDaComprare
  FROM MigliorPrezzoPerTipo M JOIN ProdottoInVendita PV ON M.TipoProdotto = PV.TipoProdotto
  WHERE MigliorPrezzo = Prezzo
  GROUP BY Barcode
```

Selezionare, per ogni TipoProdotto usato come Ingrediente della Ricetta '*Carbonara al forno*', il corrispondente ProdottoInDispensa appartenente all'Utente '*mconti*' che scade prima mostrando anche Utente, DataScadenza, TipoProdotto, Marca, Nome, QuantitaV, QuantitaNecessaria, QuantitaRimasta.
QuantitaNecessaria è la quantità di prodotto utile per la preparazione della Ricetta '*Carbonara al forno*' per 3 persone, QuantitaRimasta è la quantità del prodotto rimasta nella dispensa di '*mconti*' dopo averne consumato QuantitaNecessaria per preparare la Ricetta '*Carbonara al forno*' per 3 persone.

```
SELECT Barcode,Utente,DataScadenza,PD.TipoProdotto,Marca,Nome, PD.QuantitaV AS QuantitaDispensa,
    (I.QuantitaV/4) * 3 AS QuantitaNecessaria,
        ROUND(PD.QuantitaV-(I.QuantitaV/4) * 3 AS QuantitaRimasta
FROM ProdottoInDispensa PD JOIN Ingrediente I ON PD.TipoProdotto = I.TipoProdotto
WHERE PD.Utente = 'mconti' AND I.NomeRicetta = 'Carbonara al forno'
        AND DataScadenza = ( SELECT MIN(DataScadenza)
                             FROM ProdottoInDispensa P
                             WHERE P.Utente = 'mconti' AND P.TipoProdotto = PD.TipoProdotto)
ORDER BY PD.TipoProdotto,DataScadenza
```