Progetto My FOODPEDIA

A.A. 2015/2016

Professore Gennaro Costagliola

Autore:Adele Rispoli

Indice

1. Specifica del problema.
2. Descrizione funzionale del progetto.
3. Architettura usata.
4. Architettura software.
5. Fonti.
   5.1. Descrizione schema fonti locali in datalog.
6. Descrizione wrapper.
7. Schema globale.
8. Mapping GAV.
9. Mapping LAV.
10. Query.
   10.1. Descrizione query in datalog.
   10.2. Descrizione query in SQL.
11. Riformulazione query.
   11.1. Unfolding algorithm.
   11.2. Bucket algorithm.
12. Elenco tecnologie utilizzate.

1. Specifica del problema

Mangiare sano e praticare un corretto stile di vita sono le regole base per combattere sovrappeso e malattie cardiovascolari. Conoscere allora il giusto rapporto tra peso e calorie, tra fabbisogno del nostro organismo e consumo dei nutrienti, è la premessa di una vita sana ed equilibrata. Calcolare le calorie di quello che si sta mangiando è molto importante nell'ambito di una dieta controllata, sia essa per dimagrire o per ingrassare, ma a volte è anche semplicemente una sola curiosità da soddisfare indipendentemente dal regime alimentare. Spesso infatti conoscere l'apporto calorico di un alimento è unito più ad un consapevole desiderio di conoscere le informazioni nutrizionali degli alimenti, e tale interesse è estremamente positivo. Sapere la quantità di calorie contenute in un alimento, infatti, da un lato aiuta a bilanciare gli alimenti nella giornata così che la dieta che si sta seguendo sia corretta ed efficace, dall'altro permette di capire e quindi evitare gli errori alimentari. Il problema alla base di questo progetto è quello di fornire le giuste quantità caloriche degli alimenti più utilizzati negli Stati Uniti. Infatti oggi giorno è sempre più diffuso verificare quante calorie sono contenute in un determinato alimento per avere un giusto fabbisogno energetico e quindi energia necessaria per l'attività dell'organismo. Il sito web realizzato 'My FOODPEDIA' infatti tramite l'uso di un dataset specifico fornisce informazioni sulle calorie totali di oltre 1000 cibi comunemente mangiati con la corrispondente quantità di porzione utilizzata solitamente. Questa informazione è la chiave per aiutare i consumatori nel seguire linee dietetiche e gestire la propria alimentazione capendo quante calorie sono contenute in nei cibi consumati. Se da un lato però è forte il desiderio di controllare l'apporto calorico dei prodotti alimentari dall'altro risulta essere molto utile ritrovare ricette di un determinato alimento o comunque effettuare una ricerca per ingredienti. Per questo negli ultimi anni stanno riscuotendo un grandissimo successo programmi televisivi di cucina. Il sito realizzato risponde ad entrambe queste necessità. Oltre ad indicare le calorie contenute in ogni alimento permette al tempo stesso di ricercare una ricetta per quell'alimento indicandone il tempo necessario per la realizzazione e la difficoltà. Tali ricette sono prelevate dal sito web foodnetwork.com del canale televisivo americano Food Network in cui vengono messi in onda programmi periodici o speciali sul cibo e la cucina. Per ogni ricetta viene indicato il cuoco che ha preparato quella ricetta, alcuni consigli, il tempo totale e il tempo per la preparazione e la cottura, il rendimento cioè quante persone posso essere servite con le quantità indicate nella ricetta ed infine il livello di difficoltà.

2. Descrizione funzionale del progetto.

Il sito realizzato come spiegato nella sezione precedente dovrà da un lato fornire tutti i valori nutrizionali di un alimento o condimento e dall'altro fornire una ricetta per ogni alimento ricercata per ordine di importanza sul sito web foodnetwork.com. Quindi formalmente le funzionalità del sito realizzato sono le seguenti.

- Il sito permette di calcolare le calorie e valori nutrizionali di cibi e anche condimenti. Per ognuno di essi viene indicata la porzione di riferimento su cui sono calcolate le calorie. Quindi ad un cibo o alimento può corrispondere più di un a tabella calorica ciascuna con porzioni diverse.
- Per quanto riguarda i cibi, è possibile ricercare i condimenti in essi contenuti e ricercare una ricetta relativa a quel cibo sul sito web www.foodnetwork.com.
- Permette di ricercare tutti i cibi il cui apporto calorico è minore di un parametro indicato.

3. Architettura usata

L'architettura utilizzata l'architettura Mediator.
In questo approccio, i dati rimangono esclusivamente in fonti locali e si estraggono solo quando il sistema viene interrogato. Questo è in contrasto con l'approccio warehouse in cui i dati vengono estratti dalle fonti locali , trasformati e caricati nel warehouse prima che la query viene effettuata. In fase di query, si accede al warehouse e non alle fonti di dati.
Approcci warehouses sono in genere da preferire per le query molto complesse, ad esempio, per il data mining.Un approccio mediatore invece viene preferito poiché evita di dover propagare in tempo reale, aggiornamenti dalle fonti locali al warehouse.
Nel mediatore approccio, si inizia progettando uno schema globale (chiamato anche mediated schema) che funge da punto di ingresso unico sul quale query globali sono poste dagli utenti. UN
problema principale è quindi specificare le relazioni, vale a dire mapping semantici tra gli schemi delle fonti locali e lo schema globale. Sulla base di questi mapping, si può rispondere a query sullo schema globale utilizzando query sopra le fonti locali.

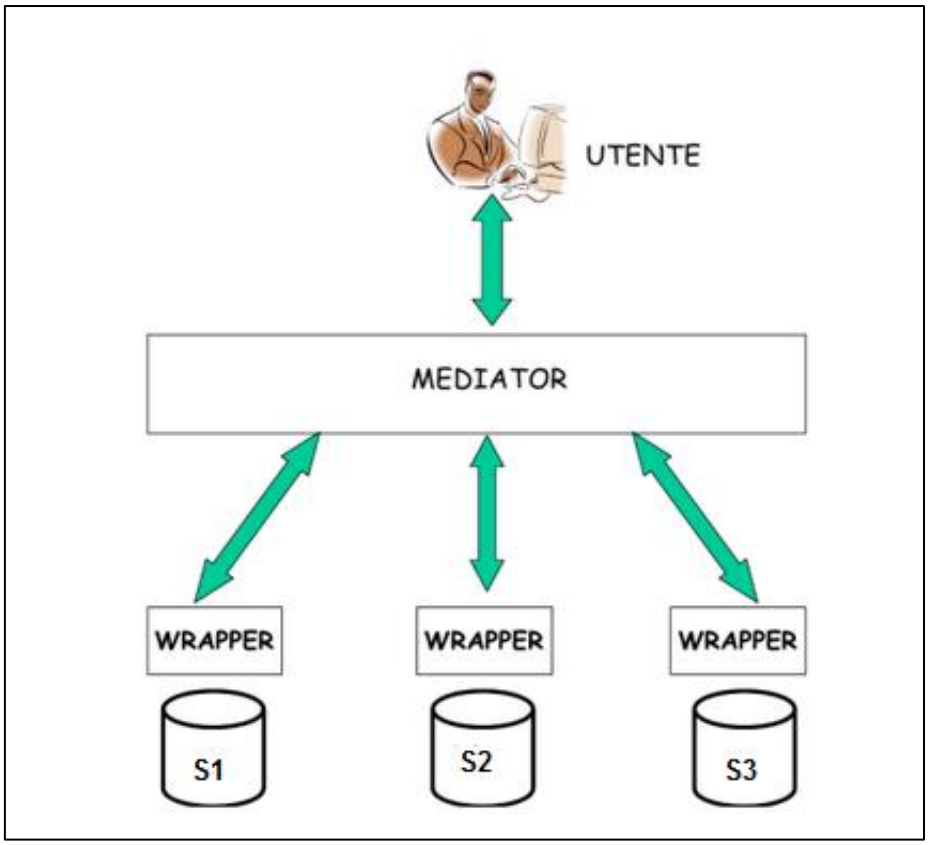

Figura 1 : Mediator architecture.

Alla base ci sono le fonti locali che possono supportare diversi tipi di query. Al di sopra ci sono programmi il cui compito è quello di comunicare con i data sources. Essi sono detti Wrapper e il loro ruolo è :

- inviare queries ai data sources;
- ricevere riposte;
- applicare trasformazioni sulle risposte in modo che queste possano essere interpretate dal processore della query.

L'utente invece interagisce con il sistema dei dati integrati tramite uno schema singolo detto Mediated schema che contiene solo gli aspetti di dominio rilevanti per l'applicazione. Nell'approccio Mediator esso è puramente uno schema logico e le relazioni tra schema globale e schema delle fonti locali viene definito tramite il mapping che può essere di tipo GAV o LAV.

4. Architettura software

L'architettura software rispecchia molto l'architettura mediator.
L'utente interagisce con l'interfaccia grafica effettuando alcune richieste. Alla sottomissione di queste verrà instanziata una classe Wrapper che fornirà i metodi opportuni per il prelievo dei dati sulle relative fonti. Questa infatti preleva i dati dalle diverse sorgenti in maniera opportuna e li trasforma in modo tale che questi possono essere processati.
Ogni pagina web quindi dovrà istanziare il Wrapper appropriato e invocare su questa i metodi opportuni.

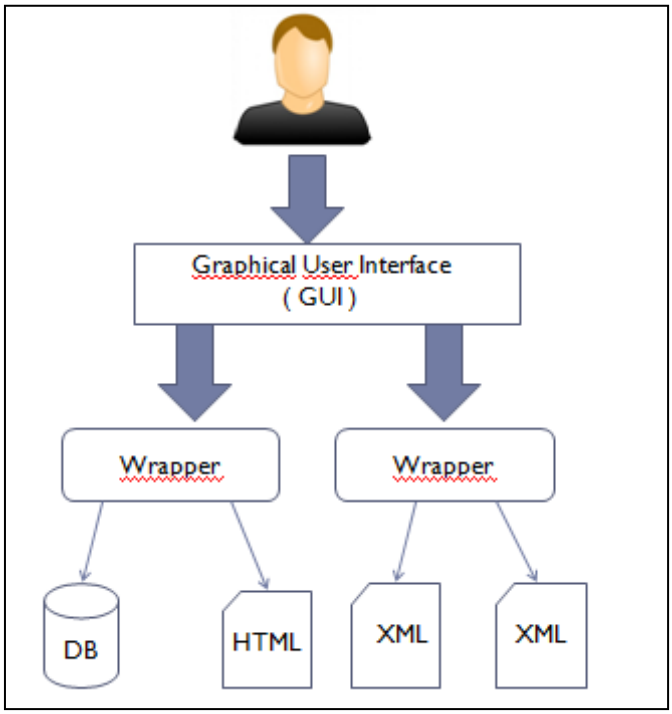

Figura 2 : Architettura software.

La figura sopra rappresenta in modo molto semplificato il modello di processo utilizzato. L'utente tramite un'interfaccia grafica molto semplice sottomette delle richieste ed in base alla richiesta effettuata viene utilizzato il wrapper opportuno che si occuperà di prelevare i dati sulle diverse fonti. Nel progetto presentato esistono 3 tipi di fonti: un database usato come cache per dati estratti dal sito web, pagine web e file xml. Per cui i wrapper sviluppati sono principalmente due: uno che estrae dati dal database e dalla pagina web e uno che invece estrae dati dai relativi file xml.

5. Fonti

Di seguito vengono descritte le fonti locali utilizzate nel progetto con il rispettivo grado di volatilità cioè se i dati sono statici o periodici.

- File xml : Food_Milk.xml / Food_No_Milk.xml
  - Contenete rispettivamente tutti i cibi con e senza latte e le relative calorie. In particolare per ogni cibo vengono indicati tutti i valori nutrizionali compreso le calorie indicandone la porzione su cui questi valori sono stati calcolati.
  - **Volatilità**: informazioni statiche

- File xml : Foods_Needing_Condiments_Table.xml
  - Contenente i cibi che possiedono condimenti. Per ogni cibo vengono elencati tutti i condimenti che sono presenti in quel cibo o che comunque possono essere aggiunti o eliminati dall'alimento selezionato.
  - **Volatilità**: informazioni statiche

- File xml : lu_Condiment_Food_Table.xml
  - Contenente tutti i condimenti e le relative calorie. In particolare per ogni condimento vengono indicati tutti i valori nutrizionali compreso le calorie indicandone la porzione su cui questi valori sono stati calcolati.
  - **Volatilità**: informazioni statiche

- Sito web : www.foodnetwork.com
  - Sito web dal quale sono stati estratti i dati relative alla ricetta scelta per ordine di importanza e a seconda dei programmi televisivi messi in onda sul canale americano di cucina Food Network per cui una ricetta per un determinato cibo può cambiare nel tempo.
  - **Volatilità**: cambi periodici

Per i dati estratti dal web è stato utilizzato un database come cache inserendo un timestamp per ciascun dato memorizzato e prevedendo il suo aggiornamento in base al grado di volatilità. La volatilità dei dati è stata gestita tramite un aggiornamento periodico dei dati sulla base della data di estrazione dei dati.

5.1. Descrizione schema fonti locali in datalog.

Di seguito viene riportata la descrizione dello schema locale di ogni fonte in datalog.

1)
Food_Milk
(Food_Code, Display_Name, Portion_Default, Portion_Amount, Portion_Display_Name, Factor, Increment, Multipler, Grains, Whole_Grains, Vegetables, Orange_Vegetables, Drkgreen_Vegetables, Starchy_vegetables, Other_Vegetables, Fruits, Milk, Meats, Soy, Drybeans_Peas, Oils, Solid_Fats, Added_Sugars, Alcohol, Calories, Saturated_Fats).

2)
Food_No_Milk
(Food_Code, Display_Name, Portion_Default, Portion_Amount, Portion_Display_Name, Factor, Increment, Multipler, Grains, Whole_Grains, Vegetables, Orange_Vegetables, Drkgreen_Vegetables, Starchy_vegetables, Other_Vegetables, Fruits, Milk, Meats, Soy, Drybeans_Peas, Oils, Solid_Fats, Added_Sugars, Alcohol, Calories, Saturated_Fats).

3)
Foods_Needing_Condiments_Table
(Survey_Food_Code, Display_Name, Cond_code, Cond_name).

4)
Condiment_Food_Table
(Survey_food_code, Display_name, Condiment_portion_size, Condiment_portion_code, Condiment_grains, Condiment_whole_grains, Condiment_vegetables, Condiment_milk, Condiemnt_meat, Condiment_oils, Condiment_solid_fats, Condiment_added_sugar, Condiment_saturated_fats, Condiment_calories).

5)
Recipe
(Dispaly_Name_Food, Name_recipe, Ingredients, Directions, Time, Difficulty, Date).

Nota : Il campo Date è usato come timestamp per poi aggiornare il database e indica la data di estrazione. Ovvero quando verrà prelevata la ricetta verrà confrontata la data di estrazione con la data odierna e se necessario verrà effettuato l'aggiornamento dei dati.

6. Descrizione wrapper.

Come detto in precedenza sono stati sviluppati due wrapper:

1) Uno per estrarre dati dai file xml;
2) Uno per estrarre dati dal database usato come cache per i dati prelevati dalla pagina web e si occupa dell'aggiornamento dei dati.

Tipo fonte → File xml

Per l'estrazione di questi dati è stata usata la libreria SimpleXML abilitata di default all'interno del motore PHP. La libreria in questione ha il grosso vantaggio di avere un'interfaccia ad oggetti molto semplice ed intuitiva che richiede poche linee di codice per accedere agli elementi interessati e mantiene intatta la struttura del file XML. In particolare nel caso degli cibi, viene fatta la ricerca su entrambi i documenti con le opportune espressioni xpath e poi si effettua la concatenazione degli array contenenti gli elementi selezionati.

```
5   public function get_calories_food($food){
6       $xmlStr1 = file_get_contents('Food_Milk.xml');
7       $xml1 = new SimpleXMLElement($xmlStr1);
8       $xmlStr2 = file_get_contents('Food_No_Milk.xml');
9       $xml2 = new SimpleXMLElement($xmlStr2);
10      $res1 = $xml1->xpath("/Food_Milk_Display_Table//Food_Display_Row[Display_Name=\"$food\"]");
11      $res2 = $xml2->xpath("/Food_NoMilk_Display_Table//Food_Display_Row[Display_Name=\"$food\"]");
12      $res=$res1+$res2;
13      //print_r($res);
14      return $res;
15  }
```

Figura 3 : Esempio di wrapper che ricerca le calorie di un alimento.

Viene creato un oggetto SimpleXMLElement passando come parametro il contenuto del file xml sotto forma di stringa. In questo modo carichiamo all'interno di un oggetto SimpleXMLElement il contenuto del file xml.
L'oggetto restituito rappresenta la root del documento xml.
Una delle funzionalità più interessanti di SimpleXML è la possibilità di interrogare l'oggetto restituito con query XPath.
Nella figura sopra riportata viene mostrata la funzione che ricerca le calorie per un determinato cibo. Per fare questo è necessario prelevare i dati sia dal file Food_Milk sia dal file Food_No_Milk e su entrambi questi effettuare la ricerca dell'elemento che ha come Display_Name uguale al nome del food richiesto con l'opportuna espressione XPath. Potrebbe infatti accadere che lo stesso cibo si trovi in entrambi i file perché avere e non avere allo stesso modo il latte e ci possono essere più risultati perché corrispondenti a porzioni diverse. Quindi dei due array ottenuti di effettua la concatenazione ottenendo così il risultato completo.

Tipo fonte→pagina web

Questa fonte è usata per effettuare la ricerca di una ricetta per un determinato cibo. Il sito web infatti permette di ricercare ricette indicando un cibo. Il risultato di tale ricerca è una lista di link a pagine web che contengono una ricetta relativa al cibo indicato. Di questa lista viene selezionato solo il primo link e dalla pagina corrispondente estratti i dati della ricetta(nome, ingredienti, preparazione, tempo e difficoltà). Quindi per fare questo è stato implementato un "piccolo" web crawler che da un indirizzo 'seed' (seme) andrà alla ricerca del collegamento ipertestuale corrispondente alla prima ricetta della lista mostrata. Una volta trovato l'indirizzo, da questa pagina vengono estratti i dati.
L'indirizzo 'seed' cioè seme da cui viene effettuata la ricerca è il seguente:

$uri = 'http://www.foodnetwork.com/search/search-results.html?searchTerm='.urlencode($nameFood).'&form=global&_charset_=UTF-8';

dove $nameFood è il nome dell'alimento di cui si vuole ricercare la ricetta.
La pagina web corrispondente a tale indirizzo contiene una lista di link a pagine web contenenti le ricette trovate. Di tutto questo elenco viene selezionato solo il primo link alla pagina web da cui saranno prelevati i dati.

```php
<?php
    $uri = 'http://www.foodnetwork.com/search/search-results.html?searchTerm='.urlencode($nameFood).'&form=global&_charset_=UTF-8';
    $get = file_get_contents($uri);
    $doc = new DOMDocument();

    libxml_use_internal_errors(TRUE); //disable libxml errors

    if(!empty($get)){ //if any html is actually returned

        $doc->loadHTML($get);
        libxml_clear_errors(); //remove errors for yucky html

        $xpath = new DOMXPath($doc);
        $row = $xpath->query('/html/body//article[@class="recipe"][1]//header//a[1]');  // prelevo href alla ricetta
    }
?>
```

Figura 4 : Ricerca del link alla pagina web con la ricetta.

Nella figura 4 viene mostrato come è stato prelevato il link con la relativa espressione XPath.
Dopo aver prelevato il link alla pagina web, si prelevano i dati relativi alla ricetta. Di seguito vengono indicate le espressioni XPath utilizzate.

```
$ingredients = $xpath->query('/html/body//div[@class="col8 ingredients responsive"]/ul/li');    // prelevo gli ingredienti
$recipe = $xpath->query('/html/body//div[@class="col10 directions"]/p');                  // prelevo la descrizione della ricetta
$time=$xpath->query('/html/body//div[@class="cooking-times"][1]');                        //prelevo il time
$difficulty=$xpath->query('/html/body//div[@class="difficulty"][1]');                     // prelevo la difficoltà
```

Figura 5 : Web scraping della ricetta.

Dopo aver estratto i dati relativi alla ricetta, si procede con l'inserimento di questi nel database usato come cache.

```php
include_once "connessioneDB.inc";
$query="DELETE FROM food WHERE name=\"$nameFood\"";
$connessione=mysql_query($query) or die (print "Class Database: Errore durante l'esecuzione della Delete");

$date=date('d-m-y');
$query = "INSERT INTO food(name,ingredients,recipe,time,difficulty,date) VALUES ('$nameFood','$ing','$rec','$t','$d','$date')";
$connessione=mysql_query($query) or die (print "Class Database: Errore durante l'esecuzione dell' Insert");
```

Figura 6 : Inserimento dati estratti nel database.

I dati estratti dalle pagine web, vengono quindi memorizzati in un database usato come cache.
Quando i dati relativi alle ricette vengono prelevati all'interno del database, il wrapper utilizzato effettuerà prima un controllo e se necessario provvederà ad aggiornare i dati prelevandoli dal sito web come descritto prima.

```php
public function control($nameFood){

    include_once "connessioneDB.inc";
    $query="SELECT date FROM food WHERE name=\"$nameFood\"";
    $result=mysql_query($query) or die (print "Class Database: Errore durante l'esecuzione dell' Insert");
    $riga=mysql_fetch_array($result);
    //echo $riga[0].'<br>';
    if(isset($riga[0])){
        $date = date_create_from_format("d-m-y",$riga[0]);
        $date_attuale=date_create_from_format("d-m-y",date("d-m-y"));
        $interval = date_diff($date, $date_attuale);
    //echo $interval->format("%a");
        if($interval->format("%a")>60){     // %a--> giorni di differenza
            aggiorna($nameFood);
        //echo "richiesto aggiornamento";
        }
        return true;
    }else
        return false;   // se la ricetta non è stata trovata nemmeno sul sito
}
```

Figura 7 : Aggiornamento dati nel database.

Ogni volta viene prelevata una ricetta dal database, il wrapper controlla se quel dato è aggiornato ovvero confronta la data di estrazione inserita come timestamp con la data odierna e se queste si discostano di 60 giorni allora viene effettuata da capo la ricerca della ricetta per quel alimento.

7. Schema globale

Di seguito verrà definito lo schema globale tramite datalog del sistema di integrazione.

1)
Food(Code, Name).

2)
Calorie(Code_Food, Portion, Calories).

3)
Food_Needing_Condiments(Code_Food, Cond_Name, Cond_Code).

4)
Condiment(Cond_Code, Cond_Name, Calories).

5)
Recipe(Name_Food, Name_Recipe, Ingredients, Directions, Time, Difficulty, Date).

Nota: per semplicità e più chiarezza non sono stati riportati tutti gli altri parametri nutrizionali come indicati più precisamente nello schema delle fonti locali ma solamente la porzione e le calorie associate per quell'alimento.

8. Mapping GAV

Di seguito viene mostrato il mapping GAV in datalog che mostra le corrispondenze tra lo schema globale e locale definendo lo schema globale in funzione di quello locale.

Food(Code, Name) ⊇ Food_Milk(Code, Name, Portion, Calories).

Food(Code, Name) ⊇ Food_No_Milk(Code, Name, Portion, Calories).

Calorie(Code_Food, Portion, Calories) ⊇ Food_Milk(Code_Food, Name, Portion, Calories).

Calorie(Code_Food, Portion, Calories) ⊇ Food_No_Milk(Code_Food, Name, Portion, Calories).

Food_Needing_Condiments(Code_Food, Cond_N_Name, Cond_N_Code) ⊇ Food_Needing_Condiments_Table(Code_Food, _ ,Cond_N_Name, Cond_N_Code).

Condiment(Cond_Code, Cond_Name, Calories) ⊇ Condiments_Food_Table(Cond_Code, Cond_Name, Calories).

Recipe(Name_Food, Name_Recipe, Ingredients, Directions, Time, Difficulty, Date) ⊇ Recipe(Name_Food, Name_Recipe, Ingredients, Directions, Time, Difficulty, Date).

9. Mapping LAV

Di seguito invece viene mostrato il mapping LAV in datalog che mostra le corrispondenze tra lo schema globale e locale definendo lo schema locale in funzione di quello globale.

Food_Milk(Code, Name, Portion, Calories) ⊆ Food(Code, Name), Calorie(Code, Portion, Calories).

Food_No_Milk(Code, Name, Portion, Calories) ⊆ Food(Code, Name), Calorie(Code, Portion, Calories).

Food_Needing_Condiments_Table(Code_Food, Display_Name, Cond_N_Name, Cond_N_Code) ⊆ Food_Needing_Condiments(Code_Food, Cond_N_Name, Cond_N_Code).

Condiments_Food_Table(Cond_Code, Cond_Name, Calories) ⊆ Condiment(Cond_Code, Cond_Name, Calories)

Recipe(Name_Food, Name_Recipe, Ingredients, Directions, Time, Difficulty, Date) ⊆ Recipe(Name_Food, Name_Recipe, Ingredients, Directions, Time, Difficulty, Date).

10. Query

Nell'applicazione sviluppata è possible poter effettuare 4 query che vengono definte come segue:

Q1: Mostrare le calorie di un determinato cibo.
Q1(Code_Food, Name_Food, Portion, Calories).

Q2: Dato un determinato cibo, mostrare i condimenti necessari e la ricetta corrispondente trovata.
Q2(Code_Food, Name_Food, Cond_Name, Name_Rec, Ingredients, Directions, Time, Difficulty, Date).

Q3: Mostrare le calorie di un condimento.
Q3(Code, Name, Portion, Calories).

Q4: Mostrare tutti i cibi con calorie minori di un parametro indicato N.
Q4(Code, Name, Portion, Calories).

10.1. Descrizione query in datalog

Q1(Code_Food, Name_Food, Portion, Calories) :-
Food(Code_Food, Name_Food), **Calorie**(Code_Food, Portion, Calories),
Name_Food="Angel food cake".

Q2(Code_Food, Name_Food, Cond_Name, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) :-
Food(Code_Food, Name_Food) , **Food_Needing_Condiments**(Code_Food, Cond_Name, _) , **Recipe**(Name_Food, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) , Name_Food="Angel food cake".

Q3(Code, Name, Portion, Calories) :-
Condiment(Code, Name, Calories) , Name = "Butter".

Q4(Code, Name, Portion, Calories) :-
Food(Code, Name), **Calorie**(Code, Portion, Calories), Calories <= 1000.

10.2. Descrizione query in SQL

Q1:

```
SELECT Food.Code, Food.Name, Calorie.Portion, Calorie.Calories
FROM Food, Calorie
WHERE Food.Name="Angel food cake" AND Food.Code=Calorie.Code_Food
```

Q2:

```
SELECT Food.Code, Food.Name,
        Food_Needing_Condiments.Cond_Name, Recipe.Name_Recipe,
        Recipe.Ingredients, Recipe.Directions, Recipe.Time,
        Recipe.Difficulty, Recipe.Date
FROM Food, Food_Needing_Condiments, Recipe
WHERE Food.Name = "Angel food cake" AND
      Food.Code: = Food_Needing_Condiments.Code_Food
      AND Food.Name = Recipe.Name_Food
```

Q3:

```
SELECT Cond_Code, Cond_Name, Calories
FROM Condiment
WHERE Cond_Name="Butter"
```

Q4:

SELECT Food.Code, Food.Name, Calorie.Portion, Calorie.Calories
FROM Food, Calorie
WHERE Food.Code = Calorie.Code_Food AND Calorie.Calories<= 1000

11. Riformulazione query

Con l'architettura Mediator, descritta nel paragrafo 3, l'utente interagisce soltanto con lo schema globale ovvero il cosiddetto Mediated schema che è uno schema puramente logico in quanto i dati sono mantenuti effettivamente nelle fonti locali; ed è questa la differenza sostanziale con invece l'architettura warehouse. Le query prima definite sono state descritte prima in datalog e poi in SQL utilizzando lo schema globale ma per poter essere eseguite queste devono essere riformulate sulle fonti locali dove sono presenti effettivamente i dati e quindi passare dallo schema globale alle fonti locali. Nel effettuare la riformulazione esistono diversi algoritmi. Se si utilizza il mapping GAV l'algoritmo utilizzato è Unfolding algorithm mentre se si utilizza il mapping LAV allora si usa il Bucket algorithm o l'inverted rule. Di seguito verrà mostrata la riformulazione della query Q2 usando l'Unfolding e il Bucket algorithm.

11.1. Unfolding algorithm

Quest' algoritmo viene applicato quando si utilizza il mapping GAV.
La riscrittura della query in query su fonti locali viene ottenuto tramite l'unfolding, cioè, sostituendo ogni atomo che può essere abbinato con testa di qualche vista, dal corpo della vista corrispondente.
Nel mapping GAV esistono due regole che corrispondono alla view Food (Food_Milk e Food_No_Milk) e un mapping per le altre. Per cui si ottengono le seguenti due query:

Q2.1(Code_Food, Name_Food, Cond_Name, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) :-
Food_Milk(Code_Food, Name,_Food _, _) ,
Food_Needing_Condiments_Table(Code_Food ,Cond_Name, _) ,
Recipe(Name_Food, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) ,
Name_Food = "Angel food cake".

Q2.2(Code_Food, Name_Food, Cond_Name, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) :-
Food_No_Milk(Code_Food, Name,_Food _, _) ,
Food_Needing_Condiments_Table(Code_Food ,Cond_Name, _) ,
Recipe(Name_Food, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) ,
Name_Food = "Angel food cake".

Il risultato della query è ottenuta calcolando Q2.1 ∪ Q2.2 .

11.2. Bucket algorithm

L'esecuzione di questo algoritmo può essere diviso in vari step.

Primo step:

- Si costruisce per ogni atomo g del corpo della query globale il suo **bucket** che ragruppa tutte le fonti locali da cui g può essere dedotto.

Q2(Code_Food, Name_Food, Cond_Name, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) :-
Food(Code_Food, Name_Food), **Food_Needing_Condiments**(Code_Food, Cond_Name, _) , **Recipe**(Name_Food, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) , Name_Food="Angel food cake".

Bucket di Food:
Food_Milk
(Code_Food,
Name_Food,v1,v2)
Food_No_Milk
(Code_Food,
Name_Food,v1,v2)

Bucket di Food_Needing_Condiments:
Food_Needing_Condiments_Table
(Code_Food,Con_Name,v3)

Bucket di Recipe:
Recipe
(Name_Food, Name_Rec,Ingredients,Dorections,Time,Difficulty,Date)

Secondo step:

- Consiste nel costruire una serie di **query candidate** che si ottengono combinando gli atomi di ogni bucket.

Dai bucket mostrati precedentemente si ottengono le seguenti combimazioni:

1) **Food_Milk** **Food_Needing_Condiments_Table** **Recipe**
2) **Food_No_Milk** **Food_Needing_Condiments_Table** **Recipe**

che corrispondono alle seguenti query candidate:

1) Q2.1(Code_Food, Name_Food, Cond_Name, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) :-
   Food_Milk(Code_Food, Name,_Food _, _) ,
   Food_Needing_Condiments_Table(Code_Food ,Cond_Name, _) ,
   Recipe(Name_Food, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) , Name_Food = "Angel food cake".

2) Q2.2(Code_Food, Name_Food, Cond_Name, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) :-
Food_No_Milk(Code_Food, Name,_Food _, _) ,
Food_Needing_Condiments_Table(Code_Food ,Cond_Name, _) ,
Recipe(Name_Food, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) , Name_Food = “Angel food cake”.

Q2.1 non è inclusa in Q2.2 e viceversa. Quindi non esiste una query che comprende l’altra per cui bisogna considerarle entrambe nel terzo step.

Terzo step:

- **Checking containement**: tra tutte le query candidate, considerare solo quelle valide o, se presente, la query che contiene massimamente tutte le altre e verificarne il suo contenimento nella query globale. Per fare questo bisogna riscrivere le query riformulate sulla fonti locali nuovamente sullo schema globale.

Nel riformulare le query su lo schema globale si ottiene in entrami i casi la seguente query:

Q(Code_Food, Name_Food, Cond_Name, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) :-
Food(Code_Food, Name_Food),
Calorie(Code_Food, v1,v2),
Food_Needing_Condiments(Code_Food, Cond_Name, _) ,
Recipe(Name_Food, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) , Name_Food=“Angel food cake”.

Semplifichismo la query eliminando la view Calorie perché inutile e quindi la query semplificata diventa:

Q(Code_Food, Name_Food, Cond_Name, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) :-
Food(Code_Food, Name_Food),
Food_Needing_Condiments(Code_Food, Cond_Name, _) ,
Recipe(Name_Food, Name_Rec, Ingredients, Directions, Time, Difficulty, Date) , Name_Food=“Angel food cake”.

La query ottenuta è identica alla query iniziale per cui non c’è bisogno di testarne l’inclusione perché sicuramente le due query sono contenute nella query globale iniziale. E quindi Q2.1 or Q2.2 è la riformulazione di Q2.

12. Elenco tecnologie utilizzate

Di seguito vengono elencate le tecnologie utilizzate per l'implementazione del progetto presentato.

- HTML e CSS : Utilizzato per l'interfaccia grafica;
- PHP : utilizzato per l'implementazione dei wrapper;
  - La libreria SimpleXML.
- XPath : Utilizzato per l'estrapolazione dei dati sia in fonti xml che in pagine web;
- XML : Utilizzato per la conservazione di dati in 4 fonti locali.