Raccolta automatica di dati metereologici dal web
A.A. 2009/2010
Elaborato
ver 1.0

Pierluigi Picciau (579308)
Relatore: Federico Filira

28 settembre 2010

Indice

Elenco delle tabelle

Capitolo 1

Stato dell'arte del recupero di informazioni sul web

1.1 Obiettivi

Il Web è una grande fonte di informazione libera e accessibile a tutti. In questa sezione verranno esaminate le problematiche relative all'acquisizione di specifiche informazioni da fonti selezionate per interesse.

Le problematiche che tipicamente si devono affrontare sono:

- valutare e soddisfare il grado di **aggiornamento** delle informazioni: a seconda dell'ambito di applicazione occorre che le informazioni acquisite siano aggiornate con più o meno frequenza;
- occorre sviluppare una **automatizzazione** del processo di acquisizione: ovvero svincolare l'azione di acquisizione delle informazioni dalla presenza dell'operatore umano.

1.2 Information Retrieval

L'information retrieval[1] (IR) è l'insieme delle tecniche utilizzate per il recupero mirato dell'informazione in formato elettronico. Per informazione si intendono tutti i documenti, i metadati, i file presenti all'interno di banche dati o nel world wide web. Il termine è stato coniato da Calvin Mooers alla fine degli anni '40 del Novecento, ma oggi è usato quasi esclusivamente in ambito informatico.
L'IR è un campo interdisciplinare che nasce dall'incrocio di discipline diverse. L'IR coinvolge la psicologia cognitiva, l'architettura informativa, la filosofia, il

[1] Letteralmente: *recupero di informazioni*

design, il comportamento umano sull'informazione, la linguistica, la semiotica, la scienza dell'informazione e l'informatica. Molte università e biblioteche pubbliche utilizzano sistemi di IR per fornire accesso a pubblicazioni, libri ed altri documenti.

Per recuperare l'informazione, i sistemi IR usano i linguaggi di interrogazione basati su comandi testuali. Due concetti sono di fondamentale importanza: query ed oggetto. Le query[2] sono stringhe di parole-chiavi rappresentanti l'informazione richiesta. Vengono digitate dall'utente in un sistema IR (per esempio, un motore di ricerca).
Una tipica ricerca di IR ha come input un comando dell'utente; poi la sua query viene messa in relazione con gli oggetti presenti nella banca dati, e in risposta il sistema fornisce un insieme di record che soddisfano le condizioni richieste.

Spesso i documenti stessi non sono mantenuti o immagazzinati direttamente nel sistema IR, ma vengono rappresentati da loro surrogati. I motori di ricerca del Web come Google e Yahoo sono le applicazioni più note ed ovvie delle teorie di Information Retrieval.

1.3 Modalità di acquisizione

In alcune applicazioni si presenta la necessità di recuperare informazioni aggiornate in modo automatizzato.

Poichè le informazioni devono essere *aggiornate*, occorre ripetere nel tempo l'acquisizione dei dati, con una frequenza che varia in funzione dell'applicazione.
Inoltre per le applicazioni che richiedono un aggiornamento in tempo reale occorre ideare una strategia per non sovraccaricare la fonte con ripetute richieste.

Essendo un'attività ripetitiva, è importante che l'acquisizione venga *automatizzata*. Se le informazioni da acquisire sono disponibili in formato puro, ovvero prive di ridondanza e di formattazione, allora l'acquisizione è immediata. Viceversa se non si dispone di informazioni pure occorre processare il dato in ingresso per interpretarlo ed estrapolare l'informazione di interesse.

1.3.1 Tecnologia push

La tecnologia di tipo *push* descrive una tipologia di comunicazione in internet, dove la richiesta per ciascuna transazione è inizializzata dal publisher (il server centrale). Diversamente, nel modello *pull* la richiesta di trasmissione è inizializzata dal ricevente (il client).

[2] interrogazioni

Uso generale

I servizi push sono spesso basati su preferenze espresse in anticipo. Questo è chiamato modello pubblica/sottoscrivi. Un client può sottoscrivere svariati canali di informazione. Non appena diventa disponibile un nuovo contenuto all'interno di questi canali, il server esegue il push delle informazioni verso gli utenti.
Conferenze sincrone e messaggistica istantanea sono un esempio tipico di servizio push. I messaggi di chat sono inviati all'utente non appena vengono ricevuti dal servizio di messaggistica.
Anche l'email è un sistema di tipo push: il protocollo SMTP sul quale è basata è un protocollo con funzionalità push. L'ultimo passo però (dal mail server al computer desktop) utilizza tipicamente un protocollo di tipo pull, come POP3 o IMAP. I client email moderni fanno sembrare istantanea questa azione grazie a un continuo polling[3] del server mail. Il protocollo IMAP include il comando IDLE, che consente al server di avvisare il client quando arriva un nuovo messaggio.

Implementazioni

HTTP server push È una tecnica per inviare dati da un web server a un web browser. Esistono diversi meccanismi per ottenere il push.
Nella maggior parte delle applicazioni push, il web server evita di terminare la connessione una volta che la risposta è stata recapitata al client. Il web server lascia la connessione aperta in modo che se un evento è ricevuto, esso può inviarlo immediatamente a uno o molteplici client. In caso contrario, una volta chiusa la connessione i dati successivi verrebbero accodati fino alla prossima richiesta del client.
La proposta WHATWG Web Applications 1.0 incluse un meccanismo per fare il push del contenuto verso il client. Ora tale meccanismo è stato standardizzato come parte di HTML5. Un'altra funzionalità relativa ad HTML è l'API WebSockets, che consente la comunicazione tra web server e client tramite una connessione TCP full-duplex.

Java pushlet Una pushlet è una tecnica originariamente sviluppata per le applicazioni web Java, ma può essere utilizzata anche in altri framework web.
In questa tecnica il server trae vantaggio dal connessioni HTTP persistenti e lascia la risposta indefinitivamente aperta (ovvero non la termina mai). Questo comportamento ha l'effetto di ingannare il browser e lo induce a restare continuamente nella modalita di caricamento anche dopo aver terminato la ricezione della pagina effettiva. Il server inoltre invia periodicamente codice javascript per aggiornare il contenuto della pagina, ottenendo così la funzionalità push.

[3] interrogazione

Usando questa tecnica il client non necessita di applet java o plugin per tenere aperta la connessione al server. Un serio problema di questo metodo è la mancanza di qualsiasi forma di controllo del server verso il timing out del client. Si rivela necessario un refresh della pagina ad ogni timeout che occorre dal lato client.

Long polling Questa tecnica è una variazione del tradizionale polling per consentire l'emulazione della tecnologia push.
Con il long polling il client richiede l'informazione al server, similmente a come farebbe con un poll normale. Se il server non dipone di informazioni per il client, invece di inviare una risposta vuota, trattiene la richiesta e aspetta che si rendano disponibili alcune informazioni. Una volta che l'informazione è disponibile (o dopo un certo timeout) viene inviata una risposta al client. Il client solitamente produrrà subito una nuova richiesta per il server, in modo che il server avrà sempre una richiesta in attesa che potrà essere utilizzata per inviare i dati in risposta a un evento.

Limitazioni della tecnologia push

L'utilizzo di una tecnologia di tipo push è possibile solo in seguito a un accordo esplicito tra publisher e client. Per ragioni puramente tecniche, è il publisher stesso che deve essere a conoscenza dei suoi client in modo da poterli avvertire non appena si rende disponibile il contenuto aggiornato.
I servizi di tipo push possono garantire un alto livello di qualità e il perfetto sincronismo tra server e client, pertanto prevedono spesso accordi di natura economica tra colui che produce l'informazione e colui che ne usufruisce.

1.3.2 Tecnologia pull

La tecnologia pull è una tecnica di comunicazione di rete che prevede una richiesta iniziale originata da un client, alla quale risponde un server. Le richieste pull sono il fondamento del network computing, dove molti client richiedono dati da server centralizzati. Il pull è usato intensivamente in internet per le richieste HTTP di pagine web dai siti. Anche moltre altre fonti web, come gli RSS, sono ottenute dal client tramite pull.

1.4 La descrizione dell'informazione: i linguaggi di Markup

Le informazioni nel web sono tipicamente descritte tramite linguaggi di markup.

Un documento si compone di:

- **struttura**: organizzazione logica del documento;
- **contenuto**: parte informativa;
- **presentazione**: aspetto grafico.

Per descrivere e separare le tre componenti di un documento vengono inserite nel documento stesso delle annotazioni (marcatori).
Affinchè sia possibile il trattamento automatico dei documenti, le annotazioni devono seguire regole sintattiche e semantiche ben precise. L'insieme di tali regole definisce il linguaggio di annotazione usato, chiamato linguaggio di markup.

I linguaggi di markup sono linguaggi formali che vengono definiti e realizzati per specificare la struttura e il formato di documenti digitali tramite l'uso di marcatori, chiamati tag.
Un tag è una parola che descrive una porzione del contenuto di un documento. L'insieme dei tag di un linguaggio di markup permette di descrivere la struttura di un documento identificando e separando i componenti logici.
L'utilizzo di linguaggi formali consenste di elaborare in modo automatico il linguaggio scritto.

SGML E' un metalinguaggio per definire linguaggi di markup. Utilizzandone i costrutti è possibile creare un numero infinito di questo tipo di linguaggi.
Un documento SGML[4] può essere facilmente letto sia da un computer che da una persona, a patto che quest'ultima conosca lo standard.
Un esempio di linguaggio derivato da SGML è l'HyperText Markup Language (HTML), ampiamente utilizzato nel web.

HTML, XML, XHTML

- **HTML**: HyperText Markup Language, è un linguaggio di markup che descrive documenti ipertestuali.
- **XML**: eXtensible Markup Language, è un sottoinsieme semplificato di SGML. Permette di definire il proprio formato di markup.
- **XHTML 1.0**: combina la forza di HTML con le potenzialità di XML.

[4] *Standard Generalized Markup Language*

1.5 Tipologie di documenti reperibili nel web

1.5.1 Pagine web

Le pagine web sono tipicamente documenti HTML/XHTML contenenti testo e riferimenti a immagini, suoni e/o contenuti multimediali.

Contenuto testuale

Il testo è presente direttamente all'interno dello stream HTML che compone la pagina. La presenza di diversi tag consente di descrivere la struttura del contenuto informativo, e sono proprio i tag, insieme ai fogli di stile (CSS[5]) a determinare come il browser deve presentare la pagina all'utente.

Contenuto multimediale

All'interno dello stream[6] HTML possono essere presenti riferimenti a file esterni di svariata natura come immagini, suoni e formati multimediali di terze parti. Il browser in questi casi scarica il contenuto aggiuntivo e lo incorpora nella pagina, con le modalità specificate nel sorgente HTML stesso. Tra i formati multimediali più diffusi vi è il formato Flash di Adobe, ormai uno standard de facto. Flash è una tecnologia che permette di creare animazioni vettoriali, anche di notevole complessità. I contenuti Flash vengono interpretati dal browser tramite un apposito plug-in, di cui esiste ovviamente anche un'implementazione prodotta da Adobe stessa.

1.5.2 Web feed

Un web feed è un formato di dati usato per distribuire agli utenti dei contenuti frequentemente aggiornati. I distributori di contenuto forniscono un feed che può essere sottoscritto dagli utenti.
Un tipico scenario di utilizzo di un web feed è il seguente: un content provider[7] pubblica un link al feed sul proprio sito, al quale gli utenti finali possono registrarsi con un programma aggregatore. Per ciascun feed sottoscritto l'aggregatore interroga il server per verificare se sono presenti nuovi contenuti ed eventualmente li scarica.
I web feed sono un esempio di tecnologia pull: gli aggregatori possono essere programmati per controllare periodicamente la presenza di aggiornamenti.

[5] *Cascading Style Sheets*
[6] flusso
[7] fornitore di contenuti

Il tipo di contenuto indicizzato dai web feed è tipicamente HTML (pagine web) o link ad esse.

Un web feed, tecnicamente, è un documento spesso basato su XML. I due formati più diffusi di feed sono RSS e Atom.
I web feed sono concepiti per facilitarne la lettura automatica oltre che quella umana. Ciò significa che i web feed possono essere utilizzati anche per trasferire automaticamente informazioni da un sito all'altro senza l'intervento umano.

1.5.3 RSS

RSS[8] è una famiglia di formati standard di web feed usati per pubblicare contenuti frequentemente aggiornati, compreso testi, audio, video.
Un documento RSS include, oltre al contenuto, dei metadati come le date di pubblicazione e l'autore.
L'utilizzo di un formato XML standard consente di pubblicare l'informazione una sola volta e di visualizzarla su molteplici dispositivi.

1.6 Acquisire i dati contenenti l'informazione: il crawler

Un *web crawler* è un software che naviga il web in modo automatico e metodico. I crawler sono chiamati anche bot, web spider o web robot.
I motori di ricerca, in particolare, usano il crawling come meccanismo per mantenere aggiornati i propri indici. I crawler sono utilizzati per mantenere una copia locale di tutte le pagine visitate, le quali saranno successivamente processate dal motore di ricerca per indicizzarle allo scopo di rendere veloci le ricerche al loro interno.
I crawler possono essere utilizzati anche per acquisire specifiche informazioni da pagine web, come ad esempio raccogliere indirizzi email (solitamente per spam).

1.7 Estrapolare l'informazione: il parsing

Per *parsing* si intende un processo di analisi sintattica di un testo composto da una sequenza di token (ad esempio parole). Il fine di tale processo è la determinazione della struttura grammaticale in riferimento a una grammatica formale.

[8] *Really Simple Syndication*

1.7.1 Parser

In informatica un *parser* è un componente che controlla la sintassi e costruisce una struttura dati di un certo input. Il parser spesso utilizza un analizzatore lessicale separato per creare i token della sequenza di ingresso. I parser possono essere programmati a mano o essere generati semi-automaticamente da qualche tool.

1.7.2 Alcune librerie per il parsing

Vengono qui riportate alcune librerie utilizzate per il parsing in vari linguaggi di programmazione.

- **(PHP) MagpieRSS**: libreria in PHP che fornisce un parser per RSS basato su XML.
- **(Python) Universal Feed Parser**: parsing di feed RSS e Atom in Python.
- **(Ruby) rss.rb**: RSS parser della libreria standard di Ruby.
- **(Python) HTML Parser**: semplice parser per HTML e XHTML, modulo della libreria standard di Python.
- **(Java) HTML Parser**: libreria in java per il parsing di HTML.

Capitolo 2

Stato dell'arte dell'organizzazione delle informazioni storiche

2.1 Obiettivi

Ogni organizzazione è dotata di un sistema informativo, che organizza e gestisce le informazioni necessarie per perseguire gli scopi dell'organizzazione stessa. Un sistema informativo non è necessariamente un sistema automatizzato o digitale, e la sua esistenza prescinde da queste due caratteristiche. In questa sezione si vuole analizzare appunto la parte automatizzata di un sistema informativo e vedere quali sono i principali sistemi per la gestione e la memorizzazione delle informazioni.

2.2 Gestione delle informazioni

Per indicare la parte automatizzata del sistema informativo viene di solito usato il termine sistema informatico.
In quest'ultimo bisogna differenziare il termine dato e informazione. I primi da soli non assumono nessun significato ma, una volta interpretati e correlati opportunamente, forniscono informazioni che consentono di arricchire la nostra conoscenza del mondo:

- **informazione**: notizia, dato o elemento che consente di avere conoscenza più o meno esatta di fatti, situazioni, modi di essere;
- **dato**: ciò che è immediatamente presente alla conoscenza, prima di ogni elaborazione; (in informatica) elementi di informazione costituiti da simboli che devono essere elaborati.

Un file consente di memorizzare e ricercare dati, ma fornisce solo semplici meccanismi di accesso e condivisione. In questo modo le procedure scritte in un linguaggio di programmazione sono completamente autonome perché ciascuna di esse utilizza un file "privato". Eventuali dati di interesse per più programmi sono replicati tante volte quanti sono i programmi che li utilizzano. Questo provoca ridondanza[1] e possibilità di inconsistenza[2].
Le base di dati risolvono questi tipi di inconvenienti gestendo in modo integrato e flessibile le informazioni di interesse. Però sono semplicemente una collezione di dati che necessita di essere gestita da un apposito sistema.

2.3 Data Base Management System

Un sistema di gestione di basi di dati è un sistema software in grado di gestire collezioni di dati che siano:

- **grandi**: possono avere dimensioni anche enormi e comunque in generale molto maggiori della memoria centrale disponibile;
- **condivise**: applicazioni e utenti diversi devono poter accedere, secondo opportune modalità, a dati comuni. In questo modo si riduce la ridondanza dei dati, perché si evitano ripetizioni, e conseguentemente si riduce la possibilità di inconsistenza;
- **persistenti**: hanno un tempo di vita illimitato a quello delle singole esecuzioni che le utilizzano.

Chiaramente, in ambito professionale, devono essere garantite anche:

- **affidabilità**: la capacità di conservare intatto il contenuto della base di dati (o di permetterne la ricostruzione) in caso di malfunzionamento hardware/software. I DBMS[3] tramite salvataggio (backup) e ripristino (recovery) gestiscono in modo controllato versioni replicate dei dati;
- **efficienza ed efficacia**: capacità di svolgere le operazioni utilizzando una politica di risparmio di tempo e di occupazione di spazio, garantendo la produttività delle attività degli utenti.

[1] presenza di dati duplicati
[2] presenza di dati che non rispecchiano le informazioni di interesse
[3] *Data Base Managment System*

2.3.1 Modelli Logici

Un modello dei dati è un insieme di concetti utilizzati per organizzare i dati di interesse e descriverne la struttura in modo che essa risulti comprensibile a un elaboratore. Ogni modello dei dati fornisce meccanismi di strutturazione, analoghi ai costruttori di tipo dei linguaggi di programmazione, che permettono di definire nuovi tipi sulla base di tipi (elementari) predefiniti e costruttori di tipo. Le strutture utilizzate da questi modelli, pur essendo astratte, riflettono una particolare organizzazione.

Modello Gerarchico Questo modello nasce alla fine degli anni 60 con l'immissione sul mercato da parte di IBM di IMS (il primo DBMS in assoluto e, appunto, gerarchico). Il nome del modello riflette la struttura sulla quale si appoggia: ogni database è diviso in archivi, a loro volta suddivisi in segmenti (o rami); infine i segmenti sono in relazione tra di loro attraverso legami padre-figlio. Si individua dunque un segmento principale (o radice) dal quale dipendono tutti gli altri segmenti figli. In virtù di questa dipendenza dal padre è possibilie fare dei rifermienti solo passando attraverso la radice, ed inoltre non è possibile, dato un figlio, risalire al padre. Chiaramente questo tipo di architettura, utilizzato per la gestione di grosse moli di dati, non è efficiente in caso di una gestione dinamica dei dati.

Modello Reticolare Il modello reticolare nasce dalla necessità di adattare il modello gerarchico a situazioni più complesse, e per questo c'è chi lo considera come un'estensione di quello gerarchico. La prima differenza consiste nella pluralità di padri che ogni nodo può avere; inoltre per questo modello esistono i normali record, e le correlazioni tra questi vengono espresse attraverso record particolari chiamati record di collegamento (*member*). Oltre ai record (normali e member) c'è una seconda struttura fondamentale chiamata *set* che permette di correlare i record per mezzo di catene di puntatori. Dunque uno schema conterrà dei record collegati da dei set.

Modello Relazionale Il modello relazionale dei dati, sviluppatosi attorno agli anni settanta, è attualmente il più diffuso e permette di definire, per mezzo del costruttore relazione (o stato di relazione o istanza di relazione), l'organizzazione dei dati in insiemi di record (tuple) a struttura fissa. Una relazione viene spesso rappresentata per mezzo di una tabella, in cui le righe rappresentano specifici record (tuple) e le cui colonne corrispondono a campi del record (attributi).

Questo modello si basa su due concetti: relazione e tabella. Infatti, dati due insiemi D1 e D2 si chiama prodotto cartesiano di D1 e D2, l'insieme delle coppie ordinate (v1, v2), tali che v1 è un elemento di D1 e v2 è un elemento di D2. Una *relazione*

matematica sugli insiemi D1 e D2 è un sottoinsieme del prodotto cartesiano di D1 x D2. Queste relazioni possono essere rappresentate graficamente in maniera espressiva sotto forma tabellare. Una tupla a questo punto è definita come:

> *Una tupla su un insieme di attributi X (x rappresenta l'insieme di attributi della relazione) è una funzione t che associa a ciascun attributo A di X un valore del dominio DOM(A)*

La grande potenza di questo modello impone anche un certo grado di rigidà. Infatti per rappresentare in modo semplice la non disponibilità di valori per un dato attributo viene inserito un particolare valore, il *valore nullo*. E' inoltre necessario evitare l'inserimento di dati sbagliati o privi di senso, e per questo esistono dei vincoli ben definiti.
I principali sono:

- Vincolo di dominio
- Vincolo di univocità
- Vincolo di integrità dell'entità
- Vincolo di integrità referenziale

Modello ad Oggetti Lo stile di programmazione moderno tende ad essere sempre più orientato verso la programmazione ad ogetti, e questo rende i DBMS ad Ogetti ideali per programmatori, che possono così sviluppare DBMS come fossero oggetti, e all'occorrenza replicarli o modificarli per crearne di nuovi. L'informazione si è evoluta nel tempo ed oggi, molto più rispetto a qualche anno fa, questa non include solo dei dati ma anche video, grafici, file audio e foto che sono considerati dati complessi. I DBMS relazionali non sono in grado di gestire in maniera efficiente questi dati. Grazie all'integrazione con i linguaggi di programmazione, il programmatote può gestire in un solo ambiente anche il DBMS ad oggetti poiché utilizza lo stesso modello di rappresentazione. Al contrario, utilizzando DBMS relazionali, i programmi che trattano dati complessi dovrebbero essere divisi in due parti: il databse e l'applicativo.

Di tutti i modelli logici elencati quello che più si è diffuso è certamente il modello Relazionale, introdotto da Edgar F. Codd. I DBMS che si appoggiano a questo modello vengono chiamati RDBMS.[4]
Una nota definizione di ciò che costituisce un RDBMS è data dalle 12 regole di

[4] *Relational Database Management System*

Codd. Tuttavia molte delle prime implementazioni del modello relazionale non erano conformi a tali regole, per cui il termine venne gradualmente cambiato fino a descrivere una più ampia classe di sistemi di basi di dati.

I requisiti minimi perchè un sistema venisse riconosciuto come RDBMS erano:

- deve presentare i dati all'utente sotto forma di relazioni (una rappresentazione a tabelle può soddisfare questa proprietà)
- deve fornire operatori relazionali per manipolare i dati in forma tabellare

2.3.2 Livelli di astrazione nei DBMS

Esiste un'architettura standardizzata articolata su tre livelli e per ognuno di questi esiste uno schema:

- **schema logico**: descrizione dell'intera base di dati per mezzo del modello logico adottato dal DBMS (modello relazionale, modello gerarchico, modello reticolare e modello a oggetti);
- **schema fisico o interno**: rappresentazione dello *schema logico* utilizzato per mezzo di strutture fisiche di memorizzazione;
- **schema esterno**: descrizione di una porzione della base di dati di interesse per mezzo del modello logico. Uno schema esterno può prevedere organizzazioni dei dati diverse rispetto a quelle utilizzare nello *schema logico*. Quindi è possibile associare a uno *schema* logico vari schemi esterni.

2.3.3 Indipendenza dei dati

L'architettura a tre livelli garantisce un'essenziale qualità ai DBMS: l'indipendenza dei dati. Questa può essere suddivisa in:

- **indipendenza fisica**: consente di interagire con il DBMS in modo indipendente dalla struttura fisica dei dati. Si possono così modificare le strutture fisiche (per esempio la modalità di gestione dei file) senza influire sulle descrizioni ad alto livello e quindi sui programmi che utilizzano i dati stessi;
- **indipendenza logica**: consente di interagire con il livello esterno della base di dati in modo indipendente dal livello logico. Per esempio è possibile aggiungere uno schema esterno senza dover modificare lo schema logico e la sottostante organizzazione fisica dei dati.

Gli accessi alla base di dati avvengono solo attraverso il livello esterno (che può coincidere con quello logico). È il DBMS a tradurre queste operazioni per i livelli sottostanti.

2.3.4 Linguaggi

Esistono diversi tipi di linguaggi che consentono di interagire con il DBMS e si distinguono in base alle loro funzioni; le due grandi categorie sono:

- **Data Definition Language** (*DDL*): linguaggi di definizione dei dati, utilizzati per definire gli schemi logici, esterni e fisici e e le autorizzazioni per l'accesso.
- **Data Manipulation Language** (*DML*): linguaggi di manipolazione dei dati, utilizzati per l'interrogazione e l'aggiornamento delle istanze di basi di dati.

Structured Query Language È un linguaggio strutturato di interrogazione *completo* perchè permette sia la definizione dei dati (DDL) che la manipolazione (DML) attraverso aggiornamenti ed interrogazioni. Questo linguaggio è stato progettato per gestire dati in un sistema di tipo relazionale. Consente di creare e modificare schemi di database, oltre a permettere l'utilizzo dei dati e la gestione degli strumenti di controllo e di accesso.
SQL utilizza dei costrutti di programmazione denominati *query* per le interrogazioni.
Creato da IBM negli anni settanta per gestire il database relazionale da loro brevettato, inizialmente prese il nome di Sequel, e nel 1986 l'ANSI[5] lo standardizzò con la sigla SQL-86.
La maggior parte delle implementazioni ha come interfaccia la classica linea di comando per l'esecuzione dei comandi, in alternativa all'interfaccia grafica.

2.3.5 Implementazioni attuali

Esistono diversi DBMS attualmente utilizzati dalle aziende di tutto il mondo. Alcuni di questi sono proprietari, altri sono di tipo Open Source. Di seguito l'elenco dei più diffusi:

DBMS proprietari

- **IBM DB2**: DB2 è un RDBMS della IBM. La sua prima versione risale al 1983 e secondo molti è stato il primo prodotto a utilizzare il linguaggio SQL.

[5] *American National Standard Institute*

- **Microsoft SQL Server**: Microsoft SQL Server è un RDBMS prodotto da Microsoft. Nelle prime versioni era utilizzato in prevalenza per basi dati medio-piccole, ma a partire dalla versione 2000 ha preso piede anche per la gestione di basi dati di grandi dimensioni.

- **Microsoft Access**: Microsoft Access è un RDBMS realizzato da Microsoft, incluso nel pacchetto Microsoft Office Professional ed unisce il motore relazionale Microsoft Jet Database Engine ad una interfaccia grafica.

- **Oracle**: Oracle è uno tra i più famosi RDBMS. La prima versione di Oracle risale al 1977, da allora sono state introdotte numerose modifiche e miglioramenti per seguire gli sviluppi tecnologici.

Attualmente DB2 e Oracle si contendono il primo posto nel mercato dei DBMS.

DBMS open source o free software

- **MySQL**: MySQL è un RDBMS composto da un client con interfaccia a caratteri e un server, entrambi disponibili sia per sistemi Unix come GNU/-Linux che per Windows, anche se prevale un suo utilizzo in ambito Unix.

- **PostgreSQL**: PostgreSQL è un completo database relazionale ad oggetti rilasciato con licenza libera (stile Licenza BSD[6]). PostgreSQL è una reale alternativa sia rispetto ad altri prodotti liberi come MySQL e Firebird SQL che a quelli a codice chiuso come Oracle o DB2. Offre caratteristiche uniche nel suo genere che lo pongono per alcuni aspetti all'avanguardia nel settore dei database.

- **Firebird SQL**: Firebird SQL è un RDBMS opensource distribuito sotto licenza IPL[7]. Supporta numerosi sistemi operativi e le principali caratteristiche di questo prodotto sono l'alto livello di conformità con gli standard SQL, la completa integrazione con molti linguaggi di programmazione e la facile installazione e manutenzione del software.

- **SQLite**: SQLite è una libreria software scritta in linguaggio C che implementa un DBMS SQL di tipo ACID incorporabile all'interno di applicazioni. È stato rilasciato nel pubblico dominio dal suo creatore, D. Richard Hipp. SQLite permette di creare una base di dati (comprese tabelle, query, form, report) incorporata in un unico file, come nel caso dei moduli Access di Microsoft Office e Base di OpenOffice.org.

[6] *Berkeley Software Distribution*
[7] *Interbase Public License*

2.4 Criteri di scelta e DBMS a confronto

Per decidere quale sia il DBMS più adatto ad ogni esigenza è necessario analizzare diversi aspetti, e non solo quelli di natura meramente economica.
Una volta definite le caratteristiche di interesse ogni DBMS preso in considerazione verrà valutato secondo una scala predeterminata.
La valutazione potrebbe quindi basarsi sulle funzionalità integrate, il tipo di licenza, la compatibilità con il sistema operativo, i tipi di dato supportati, la capacità di supportare oggetti esterni, la sicurezza, le caratteristiche e funzionalità dei tool di supporto.
Chiaramente la scelta finale ricadrà sul pacchetto che ha ottenuto il punteggio maggiore.

Per comporre la valutazione appena descritta si utilizzano delle tabelle, come le seguenti, che mettono in risalto le qualità dei diversi DBMS.

DBMS	**Costruttore**	**Ultima versione stabile**	**Licenza**
DB2	IBM	9.7 (22/04/2009)	Proprietaria
Microsoft Access	Microsoft	14 (2010)	Proprietaria
MySQL	Sun Microsystems	5.1.46 (06/04/2010)	GPL
Oracle	Oracle Corp.	11g Rel.2 (10/2009)	Proprietaria
PostgreSQL	PostgreSQL GDG[8]	8.4.4 (17/05/2010)	Free and Open Source
SQLite	D. Richard Hipp	3.6.22 (06/01/2010)	Dominio Pubblico

Tabella 2.1: Informazioni generali

DBMS	**Windows**	**Mac OS X**	**Linux**	**Unix**
DB2	Sì	Sì	Sì	Sì
Microsoft Access	Sì	No	No	No
MySQL	Sì	Sì	Sì	Sì
Oracle	Sì	Sì	Sì	Sì
PostgreSQL	Sì	Sì	Sì	Sì
SQLite	Sì	Sì	Sì	Sì

Tabella 2.2: Compatibilità Sistemi operativi

DBMS	**Dimensione Max DB**	**Dimensione Max tabella**	**Dimensione Max riga**	**Colonne per riga (max)**
DB2	512 TB	512 TB	32,677 B	1012
Microsoft Access	2 GB	2 GB	16 MB	256
MySQL	Illimitata	256 TB	64 KB	4096
Oracle	Illimitata	4 GB * dimensione blocco	8 KB	1000
PostgreSQL	Illimitata	32 TB	1.6 TB	250-1600
SQLite	32 TB	N.D.	N.D.	32767

Tabella 2.3: Limiti delle dimensioni dei dati

DBMS	**Network Encryption**	**Regole complessità password**	**Separazione ruoli utente**	**Certificato di sicurezza**
DB2	Sì	Sì	Sì	Sì (EAL4+)
Microsoft Access				
MySQL	Sì (SSL 4.0)	No	No	Sì
Oracle	Sì	Sì	Sì	Sì (EAL4+)
PostgreSQL	Sì	No	No	Sì (EAL1)
SQLite	No (Solo permessi file)	No	No	No

Tabella 2.4: Sicurezza e controllo degli accessi

Capitolo 3

Scenario di sviluppo con specifiche funzionali

3.1 Obiettivi

Si intende realizzare un pacchetto software per l'acquisizione e la memorizzazione di informazioni metereologiche.
Il compito del software consiste nel reperimento di un insieme di dati meteo, prestabiliti e relativi a specifiche località (appartenenti a province/regioni/nazioni europee) da una o più fonti web. I dati devono essere successivamente memorizzati con lo scopo di crearne uno storico.
L'ambito di applicazione di tale sistema è il settore del commercio ortofrutticolo. Il sistema dovrebbe aiutare gli operatori di questo settore a prevedere gli andamenti di mercato dei prodotti scambiati; tali andamenti sono infatti collegati alle condizioni climatiche che influenzano le colture.

3.2 Tipologia dei dati da rilevare

I dati di interesse per l'applicazione possono essere ridotti ad un insieme limitato. Le condizioni meteo da rilevare sono le seguenti:

- **temperature** massime e minime del giorno, misurate in gradi celsius;
- **precipitazioni** relative alla giornata e misurate in millimetri del totale caduto;
- **umidità relativa** dell'aria misurata in percentuale;
- **irraggiamento UV** una misura dell'irraggiamento solare espressa nell'indice UVI

3.3 Dimensione geografica dei dati da rilevare

Località italiane

Si è interessati a raccogliere i dati meteo relativi a singoli comuni italiani.
Nel caso in cui un comune fosse troppo piccolo per comparire tra le rilevazioni meteo, si intendono come accettabili i dati del capoluogo di provincia a cui tale comune appartiene. Non sono ammissibili maggiori approssimazioni.

Località europee

Il sistema dovrà raccogliere dati anche su scala europea.
Nel caso in cui una località europea fosse troppo piccola per comparire tra le rilevazioni meteo, si intendono come accettabili i dati di una delle località limitrofe.

3.4 Frequenza dei rilevamenti

La frequenza con cui i dati dovranno essere rilevati è regolare, e si è deciso essere giornaliera.
Si è stabilito pertanto che l'unità temporale minima a cui deve fare riferimento un singolo dato è la giornata, riservandosi però la possibilità di aumentare la granularità dei campionamenti qualora si rivelasse necessario.

3.5 Profondità dei dati memorizzati

Oltre ai dati relativi alla giornata odierna, si vogliono memorizzare anche i dati previsionali di ciascuna delle 4 giornate successive a quella di campionamento.
In tal modo i dati memorizzati non avranno uno sviluppo dimensionale lineare, ovvero non seguiranno solamente la scansione temporale. I dati memorizzati avranno estensione bidimensionale, comprendendo sia la scansione temporale che quella previsionale.

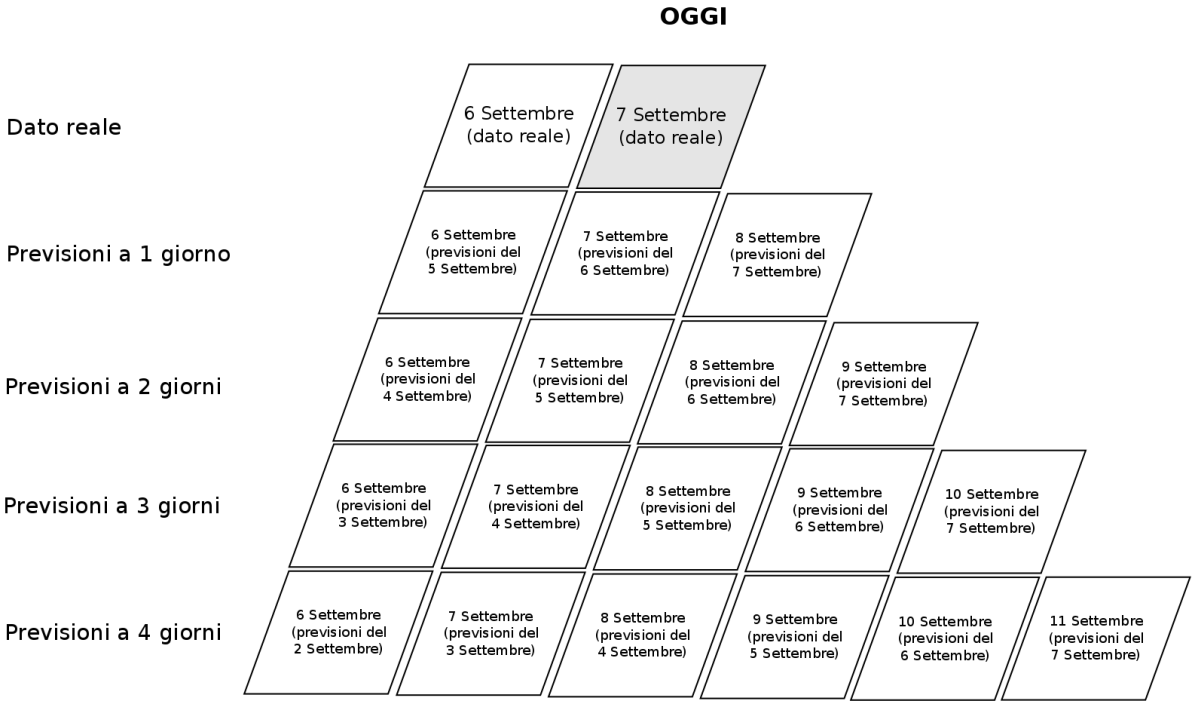


Figura 3.1: Profondità dei dati memorizzati

3.6 Dimensione dell'archivio di storicizzazione

L'archivio storico cresce indefinitamente a partire dal giorno in cui viene acceso il servizio.
Il sistema continuerà a storicizzare dati per tutta la durata della sua operatività.

3.7 Funzionalità del sistema

Oltre all'attività di acquisizione e storicizzazione dei dati, che avviene in automatico ed è trasparente all'utente, il sistema deve mettere a disposizione degli strumenti per effettuare la consultazione e semplice analisi dei dati archiviati.
Si richiede innanzitutto un'analisi a breve termine, congiunturale, riguardante quindi l'aspetto previsionale dei giorni immediatamente a seguire. Si chiede inoltre un'analisi a lungo termine, ovvero una previsione lontana nel tempo basata esclusivamente su dati storici.

Capitolo 4

Analisi e ranking delle fonti web disponibili

4.1 Obiettivi

Si vogliono ricercare e analizzare le possibili fonti di informazioni metereologiche dalle quali il sistema possa acquisire i dati di interesse per l'applicazione.
Tra le fonti analizzate esistono servizi a pagamento: qualora presenti verranno menzionati, ma nella decisione finale sulla fonte da utlizzare la scelta verrà ristretta ai soli servizi gratuiti, in quanto i costi rilevati nell'analisi risultano eccessivi per il progetto.

4.2 Panoramica delle fonti

Vengono studiati i seguenti portali web:

- *ilmeteo.it*
- *meteo.it*
- *3bmeteo.com*
- *ARPA**

4.2.1 ilmeteo.it

Figura 4.1: Sito www.ilmeteo.it

Il portale *ilmeteo.it* rappresenta uno dei siti di informazione più visitato in Italia e tra i primi in Europa[1].
Il sito web principale è accompagnato anche da una versione mobile adatta a dispositivi a piccolo schermo. Si è deciso di esaminare anche questa versione in quanto presenta un'ottima leggibilità.

Servizi a pagamento

Si analizzano per primi i servizi a pagamento erogati dal portale. Come si evince dalla seguente tabella, i dati vengono resi disponibili in formato XML, relativi a una o più località.
Il formato XML garantisce la massima semplicità di acquisizione in quanto descrive i dati strutturandoli ed eliminando ambiguità e formattazione.

[1]Fonte: Audiweb

Servizio	**Validità**	**Prezzo**
XML giornalieri e triorari, per 7gg, 1 località	1 anno	€ 125,00 + iva
XML mari e venti, 1 località	1 anno	€ 150,00 + iva
XML giornalieri e triorari, per 7gg, tutte le province	1 anno	€ 2.500,00 + iva
XML giornalieri e triorari, per 7gg, tutti i comuni	1 anno	€ 9000,00 + iva

Tabella 4.1: Costo dei servizi offerti da *ilmeteo.it*

Gli ultimi due pacchetti (province e comuni) sono gli unici di effettivo interesse per l'applicazione in esame. Dalla tabella è evidente che si tratta di costi annuali relativamente elevati.
Il vantaggio principale di affidarsi a una soluzione a pagamento consiste nel vedersi garantito un livello minimo di qualità, definito contrattualmente e pertanto esigibile per legge.

Servizi gratuiti

Come già esposto, *ilmeteo.it* mette a disposizione un portale web, un sito mobile e anche un'applicazione mobile per piattaforma iOS[2].
Vengono offerte previsioni orarie e triorarie per tutti i comuni italiani e per le principali città europee.
Queste tre incarnazioni di *ilmeteo.it* si differenziano solamente per la modalità di presentazione dei dati; quest'ultimi invece sono gli stessi, anche se ogni piattaforma visualizza un insieme di dati parziale. L'unica piattaforma che visualizza tutti i dati è il portale web.
La versione web risulta inoltre l'unica a essere completa, infatti presenta tutti i dati necessari all'applicazione che si sta sviluppando. La versione mobile è completa ad eccezione del dato relativo all'irradiamento UV.

Applicazione per piattaforma iOS

Il portale *ilmeteo.it* mette a disposizione degli utenti un'applicazione gratuita per la piattaforma iOS.
L'applicazione, attraverso un'interfaccia ottimizzata per piccoli schermi touch screen, presenta una selezione dei dati pubblicati nel portale web.
L'utilizzo di questa applicazione come fonte da cui acquisire i dati richiede il suo reverse engineering (ad esempio facendo lo sniffing del traffico TCP/IP[3]). Si decide pertanto di scartare a priori la possibilità di utilizzare l'applicazione come fonte,

[2] Apple iPhone, iPad e iPod Touch
[3] *Transmission Control Protocol / Internet Protocol*

considerato anche che tutti i dati visualizzati dall'applicazione sono disponibili nel portale web.

Figura 4.2: Applicazione *ilmeteo.it* per piattaforma iOS

4.2.2 3bmeteo.com

Figura 4.3: Sito www.3bmeteo.com

Il sito web principale di *3bmeteo.com* è accompagnato anche da una versione mobile adatta a dispositivi a piccolo schermo. Si è deciso di esaminare anche questa versione in quanto presenta un'ottima leggibilità.
Vengono offerte previsioni orarie e previsioni divise in fasce (mattina-pomeriggio-sera-notte) per tutti i comuni italiani e per le principali città europee.
3bmeteo.com mette a disposizione un portale web, un sito mobile e anche un'applicazione mobile per piattaforma iOS (iPhone, iPad e iPod Touch).
Le tre versioni di *3bmeteo.com* si differenziano solamente per la modalità di presentazione dei dati; quest'ultimi invece sono gli stessi, anche se non tutti i dati sono visualizzati su tutte le piattaforme.
La versione web risulta completa ad eccezione del dato relativo all'irraggiamento UV. La versione mobile invece è carente di molte informazioni: essa presenta

soltanto la temperatura e una descrizione sintetica dello stato meteo (ad esempio: *Cielo poco o parzialmente nuvoloso*).

4.2.3 meteo.it

Figura 4.4: Sito www.meteo.it

Il portale *meteo.it* offre previsioni e stato meteo per diverse località italiane, presentando il contenuto informativo sottoforma di numerose mappe interattive e filmati. Il contenuto testuale è molto scarso e poco strutturato.
Le previsioni sono effettuate su fasce orarie piuttosto ampie e approssimativamente definite (*Mattina, Pomeriggio, Sera, Notte*).

4.2.4 ARPA*

Per *Arpa** qui si intendono i siti delle Agenzie Regionali per la Protezione dell'Ambiente. Ad esempio l'*Arpav* è l'agenzia regionale del Veneto.
La maggior parte di queste agenzie elabora delle previsioni meteo su scala regionale, pubblicandole successivamente all'interno del proprio sito istituzionale.
In genere le previsioni elaborate vengono presentate sottoforma di riepilogo delle generali condizioni meteo, peccando quindi in quanto a precisione sui singoli dati.
A questo problema si aggiunge la frammentazione dei vari siti regionali, che renderebbe necessaria l'analisi di ciascuno di essi al fine di coprire l'intero territorio nazionale.
Viste le precedenti considerazioni, si è deciso di scartare le *ARPA** dalle possibili fonti.

4.3 Criteri per il ranking delle fonti

In questa sezione si intende classificare le fonti precedentemente individuate per stabilire la migliore o le migliori da utilizzare nell'applicazione.
Risulta evidente che non esiste una classificazione unica: a seconda del parametro che si prende in considerazione si ottengono classifiche differenti.
Si è deciso di considerare i seguenti aspetti:

- **Completezza** Per completezza si intende la quantità di informazioni presenti, ma relativamente alle sole informazioni di interesse. Una fonte può essere considerata completa se rende disponibili tutti i dati che si è scelto di rilevare nel terzo capitolo.

- **Attendibilità** Per attendibilità si intende la qualità dell'informazione presente, intesa come il grado di corrispondenza con il dato reale. Una fonte può essere considerata tanto più attendibile quanto più i suoi dati sono vicini al dato reale. Per valutare il grado attendibilità è necessario disporre di un archivio storico già popolato, e quindi valutare con esso di quanto le previsioni si siano discostate mediamente dal valore reale registrato. Non disponendo di tale archivio, si è deciso di valutare l'attendibilità in base alle referenze dei vari servizi. Ad esempio, *ilmeteo.it* è stato valutato come molto attendibile in quanto ha tra i suoi clienti *RAI Radio Televisione Italiana SPA*, *Autostrade per l'Italia Spa*, *Università degli Studi di Padova, Dipartimento di Ingegneria Idraulica.*

- **Formato dei dati** Come formato dei dati si intende valutare la facilità con cui le informazioni pubblicate sono acquisibili dal sistema.

4.4 Riepilogo dei punteggi

Fonte	**Completezza**	**Attendibilità**	**Formato dati**
ilmeteo.it	++	++	+
ilmeteo.it (versione mobile)	-	++	++
3bmeteo.com	-	+	+
3bmeteo.com (versione mobile)	- -	+	++
meteo.it	-	+	- -

Tabella 4.2: Ranking delle fonti

4.5 Scelta della fonte

La fonte che presenta le migliori caratteristiche di completezza, attendibilità e formato dati è *ilmeteo.it*, nella sua versione come portale web. In particolare è attualmente l'unica fonte che riporta il dato sull'irraggiamento UV.
Si decide quindi di utilizzare il sito web `www.ilmeteo.it` come fonte primaria di acquisizione dati. Eventualmente potrebbe essere considerata la fonte *3bmeteo.com* come fonte ausiliaria di supporto, anche se il livello di completezza della fonte principale è tale da non richiedere integrazioni.

Capitolo 5

Progettazione di un crawler per acquisizione dati automatica

5.1 Obiettivi

Si intende progettare un crawler automatico per acquisire, ripetutamente nel tempo, i dati meteo necessari all'applicazione che si sta sviluppando.
Il crawler deve occuparsi inoltre della memorizzazione dei dati acquisiti in un database.

5.2 Analisi della struttura della fonte

Come fonte si è scelto di utilizzare il portale *ilmeteo.it* . Le motivazioni di tale scelta sono esposte nel capitolo 4.

La tipica pagina che contiene i dati da estrarre è composta da 2 frame. Il frame interessante è quello identificato da un URL di questo genere: `http://www.ilmeteo.it/portale/meteo/previsioni1.php?citta=Mira&c=4093&g=1`
Questo indirizzo, ad esempio, identifica le previsioni orarie per il comune di Mira (VE).

Il frame, una volta renderizzato da un browser, si presenta come una tabella contenente i dati meteo orari disposti uno per riga.

Comune di **Mira** (VE) - CAP 30034 — Casa.it - Trovala a Mira

Meteo Giornaliero | Venerdì 20 | **Sabato 21** | Domenica 22 | Lunedì 23 | Martedì 24 | Mercoledì 25 | Giovedì 26 | Fino al 3 Settembre» novità

▸Previsioni Triorarie ▸**Previsioni Orarie** — Bollettino PDF — Opzioni — Altri dati Meteo

Ora	Tempo	T (°C)	Vento (km/h)	Precipitazioni	Percepita	Umidità	UV	Quota 0°C
24:00	sereno	21.1°	NW 3 debole	-- assenti --	22°C	93 %	0	4100m
01.00	sereno	20.5°	NW 4 debole	-- assenti --	21°C	94 %	0	4110m
02.00	sereno	20°	NW 4 debole	-- assenti --	21°C	95 %	0	4110m
03.00	sereno	19.6°	NW 5 debole	-- assenti --	20°C	96 %	0	4100m
04.00	sereno	19.2°	NW 6 debole	-- assenti --	20°C	96 %	0	4100m
05.00	sereno	18.8°	NW 7 debole	-- assenti --	19°C	97 %	0	4090m
06.00	sereno	18.5°	NW 7 / max 8 debole	-- assenti --	19°C	97 %	0	4090m
07.00	sereno	18.8°	NW 7 debole	-- assenti --	19°C	97 %	0.1	4080m
08.00	sereno	21.6°	NW 6 / max 8 debole	-- assenti --	22°C	90 %	0.7	4080m
09.00	sereno	24.2°	NW 7 / max 8 debole	-- assenti --	25°C	80 %	1.9	4110m
10.00	sereno	26.5°	NNW 7 / max 8 debole	-- assenti --	28°C	69 %	3.5	4140m
11.00	sereno	28.2°	N 8 / max 9 debole	-- assenti --	30°C	60 %	5.1	4180m
12.00	sereno	29.5°	NE 10 / max 12 debole	-- assenti --	31°C	55 %	6.3	4210m
13.00	sole e caldo	30.3°	NE 13 / max 16 moderato	-- assenti --	32°C	51 %	6.9	4250m
14.00	sole e caldo	30.8°	ENE 15 / max 19 moderato	-- assenti --	33°C	49 %	6.7	4290m
15.00	sole e caldo	31°	ENE 16 / max 20 moderato	-- assenti --	33°C	48 %	5.6	4330m
16.00	sole e caldo	31°	ENE 16 / max 21 moderato	-- assenti --	33°C	49 %	4.1	4350m
17.00	sole e caldo	30.7°	ENE 15 / max 21 moderato	-- assenti --	32°C	49 %	2.5	4360m
18.00	sole e caldo	30.1°	ENE 14 / max 20 moderato	-- assenti --	32°C	51 %	1	4370m
19.00	sereno	29.1°	ENE 10 / max 18 moderato	-- assenti --	31°C	58 %	0.1	4360m
20.00	sereno	26.1°	NE 8 / max 17 debole	-- assenti --	28°C	73 %	0	4340m
21.00	sereno	24.7°	NNE 9 / max 17 debole	-- assenti --	25°C	77 %	0	4320m
22.00	sereno	23.7°	NNE 9 / max 18 debole	-- assenti --	24°C	80 %	0	4330m
23.00	sereno	22.9°	NNE 10 / max 19 debole	-- assenti --	23°C	84 %	0	4340m
24:00	sereno	22.1°	NNE 10 / max 21 moderato	-- assenti --	23°C	87 %	0	4350m
Medie climatiche mese di Agosto		min 17° max 27°	SSE 9 debole	83 mm (cumulati)	min 16° max 27°	72 %	n/d	n/d

Aggiornamento del **20/08/10 10.00** - Prossimo: 20/08/10 14.00

Meteo Mira

Soleggiato e caldo. Vento da Est-Nord-Est con intensità di 15 km/h. Raffiche fino a 21 km/h. Temperature: 19°C la minima e 31°C la massima. Quota 0°C a 4200 metri.

SOLE - Sorge: 6:19, Tramonta: 20:08 **LUNA** - Leva: 18:27, Cala: 2:54 - Gibbosa crescente

Dati geografici - Lat: 45.44° Lon: 12.14° Alt: 6m s.l.m. Abitanti: 38434 • Mappa

Figura 5.1: Pagina web: previsioni orarie per il comune di Mira (VE)

L'indirizzo punta alla pagina *previsioni1.php*. Si tratta di uno script php: sperimentalmente si è constatato che accetta i parametri *citta*, *c* e *g*.
In seguito a un semplice reverse engineering con approccio black box[1] si è stati in grado di comprendere il ruolo dei singoli parametri.

```
previsioni1.php?citta=XXXXXXXX&c=XXXX&g=X
```

- *citta*: è un parametro di tipo stringa, sembra non avere alcun effetto sull'esecuzione dello script.
- *c*: è un parametro numerico intero, identifica la località.
- *g*: è un parametro numerico intero, seleziona il giorno di cui si richiedono le previsioni, con la convenzione che il valore *1* significa *domani*, il valore *2* significa *tra due giorni* e via così fino al valore massimo (*6*).

5.2.1 Sorgente HTML

Il crawler deve estrapolare le informazioni richieste partendo dal sorgente HTML. Qui se ne riporta un estratto.

```
[ ... ]

<tr>
    <th> Ora</th>
    <th> </th>
    <th>Tempo</th>
    <th>T (&deg;C)</th>
    <th></th>
    <th>Vento (km/h)</th>
    <th>Precipitazioni</th>
    <th><span id="c3a-47513" ><acronym style="cursor:help" title
        ="Indice di calore">Percepita</acronym></span><span id="
        c3b-47513" class="hdata"><acronym style="cursor:help"
        title="Wind Chill - Temperatura percepita per effetto
        del vento">W. chill</acronym></span></th>
    <th><span id="c2a-38668" >Umidit&agrave;</span><span id="c2b
        -38668" class="hdata">Pressione</span></th>
```

[1] Nel reverse engineering con approccio black box i sistemi da analizzare sono osservati senza esaminare la loro struttura interna. Si studiano invece le risposte a determinati input di prova.

```
    <th><span id="c1a-42413" ><acronym style="cursor:help" title
      ="Indice␣radiazioni␣ultraviolette.␣Valori␣da␣0␣a␣10">UV<
      /acronym></span><span id="c1b-42413" class="hdata">
      Visibilit&agrave;</span></th>
    <th><span id="c4a-11584" >Quota 0&deg;C</span><span id=
      "c4b-11584" class="hdata"><acronym style="cursor:help"
      title="Probabilit&agrave;␣di␣grandine">Grandine</acronym
      ></span></th>
</tr>

<tr class="dark">
       <td class="f">24:00</td><td><img src="http://www.ilmeteo
         .it/portale/meteo/img/s101.gif" border="0" align="
         middle" alt="" /></td>
    <td>sereno  </td><td>21.1&deg;</td>
    <td><img src="http://www.ilmeteo.it/portale/meteo/img/wind1h
      -nw.gif" border="0" align="middle" alt="NW"/></td>
    <td><acronym style="cursor:help" title="1.4␣nodi">NW 3<
      /acronym><br /><span class="descri">debole</span></td>
    <td class="pl5"> <span class="descri">-- assenti --</span></
      td>
    <td><span id="c3a-63566" >22&deg;C</span><span id="c3b-63566
      " class="hdata">21 &deg;C</span></td>
    <td><span id="c2a-14191" >93 %</span><span id="c2b-14191"
      class="hdata">1021mb</span></td>
    <td><span id="c1a-78362" >0</span><span id="c1b-78362" class
      ="hdata">&gt;10km<br/><span class="descri">buona</span><
      /span></td>
    <td><span id="c4a-40584" >4100m</span><span id="c4b-40584"
      class="hdata">0%<br /><span class="descri"></span></span
      ></td>
</tr>

[ ... ]
```

5.3 Frequenza delle acquisizioni

Considerata la struttura della fonte (dati meteo con scansione oraria) si è stabilita la seguente frequenza di acquisizione:

- **Ogni ora:** Si acquisisce il singolo dato orario attuale che rappresenta il dato reale misurato e definitivo.

Indirizzo di acquisizione:
`http://www.ilmeteo.it/portale/meteo/previsioni1.php?c=XXXX`

- **Ogni giorno:** si acquisicono i dati previsionali di tutte le 24 ore, riferiti ai 4 giorni successivi a quello di campionamento.
  Indirizzi di acquisizione:
  `http://www.ilmeteo.it/portale/meteo/previsioni1.php?c=XXXX&g=1`
  `http://www.ilmeteo.it/portale/meteo/previsioni1.php?c=XXXX&g=2`
  `http://www.ilmeteo.it/portale/meteo/previsioni1.php?c=XXXX&g=3`
  `http://www.ilmeteo.it/portale/meteo/previsioni1.php?c=XXXX&g=4`

5.4 Piattaforma di sviluppo

5.4.1 Crawler: Python

Figura 5.2: Logo di Python

Python è innanzitutto un linguaggio di script pseudocompilato. Questo significa che, similmente a Perl ed a Tcl/Tk, ogni programma sorgente deve essere pseudocompilato da un interprete. L'interprete è un normale programma che va installato sulla propria macchina e si occuperà di interpretare il codice sorgente e di eseguirlo.
Il principale vantaggio di questo sistema è la portabilità: lo stesso programma potrà girare su una piattaforma Linux, Mac o Windows purché vi sia installato l'interprete.
Python è un linguaggio orientato agli oggetti. Supporta le classi, l'ereditarietà e si caratterizza per il binding dinamico. La memoria viene gestita automaticamente e non esistono specifici costruttori o distruttori; inoltre esistono diversi costrutti per la gestione delle eccezioni.
Un altro importante elemento per inquadrare Python è la facilità di apprendimento. In questo ambito gioca un ruolo fondamentale la struttura aperta del

linguaggio, priva di dichiarazioni ridondanti e estremamente simile ad un linguaggio parlato.
L'indentazione perde il suo ruolo inteso come stile di buona programmazione per facilitare la lettura del codice, e diventa parte integrante della programmazione che consente di suddividere il codice in blocchi logici.

5.4.2 Librerie per il parsing di HTML/XML

Software	**Licenza**	**Piattaforma**	**Versione Python**
Beautiful Soup	Python License	qualsiasi (Python puro)	2.3 - 2.6/3
Mechanize	BSD	qualsiasi (Python puro)	2.4 - 2.7

Tabella 5.1: Librerie per il parsing di HTML/XML

La scelta sulla libreria da utilizzare per sviluppare il crawler è ricaduta su *Beautiful Soup*, per la ricca libreria di funzioni che mette a disposizione e la facilità di sviluppo.
Beautiful Soup è un parser in python per HTML/XML, progettato per essere utilizzato nello screen-scraping. Queste sono le caratteristiche che lo rendono potente:

- **Tollerante verso un cattivo markup:** gestisce un parse tree[2] che rappresenta il documento originale.
- **Fornisce pochi e semplici metodi in stile python** per la navigazione, la ricerca e la modifica del parse tree: è un toolkit generico per sezionare un documento ed estrarre cià di cui si ha bisogno. Non occorre scrivere un parser per ogni applicazione.
- **Converte automaticamente i documenti in Unicode:** non occorre preoccuparsi della codifica del testo, a meno che *Beautiful Soup* non riesca a rilevarla automaticamente e il testo non ne specifichi una.

5.5 Portabilità del crawler

Il linguaggio di sviluppo scelto (nella fattispecie *python*) garantisce un'elevata portabilità. Il crawler potrà essere eseguito su tutte le piattaforme per le quali è disponibile l'interprete python. Queste piattaforme comprendono le tre ben note:

[2]Un parse tree o albero sintattico (concreto) è un albero che rappresenta la struttura sintattica di una stringa in accordo a determinate forme grammaticali.

Unix/Linux, *Windows*, *Mac*.
Anche *Beautiful Soup* non pone particolari problemi alle caratteristiche di portabilità del crawler: è una libreria scritta in python puro e pertanto richiede unicamente la presenza dell'interprete per essere eseguita.

5.6 Codice sorgente del crawler

Qui si propone il codice sorgente del crawler relativo alle funzioni di parsing delle pagine web contenenti le informazioni.

```
import urllib
from BeautifulSoup import BeautifulSoup

[ ... ]

file = urllib.urlopen("http://www.ilmeteo.it/portale/meteo/
    previsioni1.php?c="+str(location)+"&g="+str(previsione))
soup = BeautifulSoup(file)
table = soup.find('table', {'class': 'datatable'})
for i in range(3,51,2):
    orario = int(table.contents[i].contents[0].contents[0][:-3])
    temperatura = float(table.contents[i].contents[3].contents
        [0].replace('&deg;',''))
    if (len(table.contents[i].contents[6])==2 and table.contents
        [i].contents[6].contents[1].contents[0]=="--␣assenti␣--"
        ):
         precipitazioni = 0
    else:
         precipitazioni = float(table.contents[i].contents[6].
             contents[0].replace(' mm','').replace('<',''))
    umidita = int(table.contents[i].contents[8].contents[0].
        contents[0].replace('␣%',''))
    try:
         UV = float(table.contents[i].contents[9].contents[0].
             contents[0])
    except TypeError:
         UV = float(table.contents[i].contents[9].contents[0].
             contents[0].contents[0].contents[0].replace(' '
             ,''))

[ ... ]
```

Capitolo 6

Progettazione dell'interfaccia di consultazione

6.1 Obiettivi

Oltre alle funzioni di acquisizione dei dati, che sono automatiche e trasparenti all'utente per opera del crawler progettato nel capitolo 5, il sistema deve fornire anche un'interfaccia di consultazione.
L'interazione tra utente finale e sistema avverrà esclusivamente tramite l'interfaccia che verrà progettata in questo capitolo.
Si è deciso di sviluppare un applicativo web-based quale interfaccia. I vantaggi di questa scelta sono:

- **Multiutenza**: siccome l'applicativo nasce con architettura client-server, è facile realizzarlo con il supporto per più utenti, visto che tutta la logica è accentrata su server.
- **Interfaccia omogenea**: l'interfaccia utente viene costruita tramite browser web e pertanto diverse applicazioni condividono una stessa interfaccia utente con indubbi vantaggi sui tempi di apprendimento da parte dell'utente finale.
- **Multipiattaforma**: poiché la logica applicativa è ospitata su server e le postazioni client necessitano solo di un browser web, l'applicativo può essere utilizzato da qualsiasi client dotato di browser web: Windows, MacOs, Linux/Unix, PDA e telefoni cellulari.

6.2 Funzioni dell'interfaccia

Si distinguono tre classi di funzionalità: amministrazione, presentazione, analisi.

6.2.1 Funzioni di amministrazione

- **Inserimento/rimozione di una località** nella lista delle località da monitorare.
- **Inserimento/rimozione di un fornitore** nella lista dei fornitori.

6.2.2 Funzioni di presentazione

- **Visualizzazione panoramica** delle località monitorate.
- **Visualizzazione panoramica** dei fornitori.
- **Visualizzazione in dettaglio** dei dati relativi a una giornata e a una località.

6.2.3 Funzioni di analisi

- **Previsione a breve termine**, basata sui dati previsionali già acquisiti in merito ai 4 giorni futuri;
- **Previsione a lungo termine**, basata sull'archivio storico.

6.3 Piattaforma di sviluppo

6.3.1 Sistema Operativo: GNU/Linux

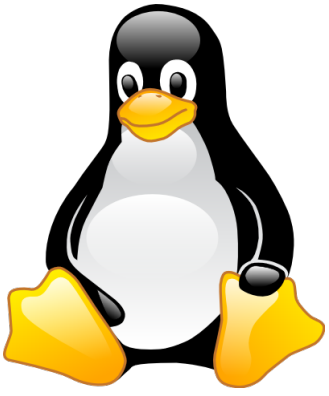


Figura 6.1: Tux: mascotte e logo di Linux

GNU/Linux è un sistema operativo. È libero, distribuito con licenza GNU GPL, di tipo Unix (o unix-like) costituito dall'integrazione del kernel Linux con elementi

del sistema GNU e di altro software sviluppato con licenze libere.
Linux è il nome del kernel sviluppato da Linus Torvalds a partire dal 1991 che, integrato con i componenti già realizzati dal progetto GNU (compilatore gcc, libreria Glibc e altre utility) e da software di altri progetti, è stato utilizzato come base per la realizzazione dei sistemi operativi open source e delle distribuzioni che vengono normalmente identificate con lo stesso nome.
Molto conosciuto nell'ambito server, Linux gode del supporto di società come IBM, Sun Microsystems, Hewlett-Packard, Red Hat e Novell ed è usato come sistema operativo su una gran varietà di hardware; dai computer desktop ai supercomputer, fino a sistemi embedded come cellulari e palmari, e dispositivi di rete.

Distribuzione: Archlinux

Figura 6.2: Logo di Archlinux

Arch Linux è una distribuzione[1] linux non specializzata che può essere modellata per fare praticamente di tutto. È veloce, leggera, flessibile e molti dei suoi componenti interni sono abbastanza semplici da capire e mettere a punto: ciò rende Arch Linux una buona distribuzione su cui fare esperienza.
Arch Linux usa pacchetti ottimizzati per architettura i686, dando così migliori performance rispetto alle distribuzioni ottimizzate per i386. Questo significa che Arch Linux funziona solo su processori di classe Pentium II o superiori.
Arch Linux usa Pacman come gestore dei pacchetti. Pacman abbina un formato semplice di pacchetti binari ad un sistema facile da usare per la compilazione di software, a partire dai sorgenti, permettendo agli utenti di gestire e personalizzare con semplicità i propri pacchetti, siano essi pacchetti ufficiali Arch Linux o pacchetti autoprodotti dall'utente stesso. Il sistema a repository consente agli utenti di compilare e mantenere i propri repository di pacchetti.
Pacman può tenere aggiornato un sistema, sincronizzando le liste dei pacchetti con il server principale, rendendo il lavoro di manutenzione molto semplice per l'amministratore di sistema. Questo modello client-server permette anche di scaricare

[1]Una distribuzione Linux, detta gergalmente anche distro, è una distribuzione software che include un kernel Linux e un insieme variabile di altri strumenti e applicazioni software, siano esse freeware, open source o commerciali.

e installare pacchetti con un semplice comando, completi di tutte le dipendenze richieste (similmente ad apt-get di Debian).
In definitiva, Arch Linux è una distribuzione adattabile, progettata per soddisfare le necessità dell'utente linux competente.

6.3.2 Web server: Apache

Figura 6.3: Logo di Apache

Come server HTTP si è scelto di utilizzare Apache, rilasciato sotto l'omonima licenza opensource Apache License.
Apache rappresenta lo standard de facto dei web server: il grande successo di diffusione di questo software è l'indicatore più chiaro della qualità e dell'affidabilità di questo prodotto: secondo un'indagine Netcraft del 2005, su 75 milioni di siti web, circa 52 milioni utilizzavano Apache, ad ottobre 2006 il numero è salito a 60 milioni (69,32% del totale).

6.3.3 Linguaggio di scripting: PHP

Figura 6.4: Logo di PHP

PHP (acronimo ricorsivo di PHP: Hypertext Preprocessor, preprocessore di ipertesti) è un linguaggio di scripting interpretato, con licenza open source e libera (ma incompatibile con la GPL), originariamente concepito per la programmazione web, ovvero la realizzazione di pagine web dinamiche.
Attualmente è utilizzato principalmente per sviluppare applicazioni web lato server ma può essere usato anche per scrivere script a linea di comando o applicazioni standalone con interfaccia grafica.
L'elaborazione di codice PHP sul server produce codice HTML da inviare al browser dell'utente che ne fa richiesta.

6.3.4 Librerie grafiche: phpgraphlib

Figura 6.5: Logo di phpgraphlib

PHPGraphLib è una libreria PHP leggera per la creazione di grafici sottoforma di immagini PNG. PHPGraphLib è gratuita da utilizzare per uso personale e può anche essere usata a scopo commerciale, con un costo contenuto.
PHPGraphLib è una classe per PHP 4.3+ con potenti caratteristiche di personalizzazione. Consente di genereare grafici a barre, linee o torta.

6.3.5 Portabilità dell'interfaccia

Si distingue la portabilità lato client e lato server.

6.3.6 Portabilità lato client

L'utilizzo di un applicativo web based quale interfaccia assicura la massima portabilità. L'interfaccia è utilizzabile da qualsiasi dispositivo dotato di web browser, quindi non solo computer ma anche tablet, palmari, telefoni cellulari.
Per aumentare l'usabilità è possibile decidere di implementare, in futuro, una versione dell'interfaccia web adatta a dispositivi con schermi ridotti.

6.3.7 Portabilità lato server

L'interfaccia è stata implementata su piattaforma LAPP[2]. Avendo utilizzato solo funzioni della libreria standard di Php e sintassi SQL standard, l'interfaccia è facilmente portabile su piattaforma LAMP[3] o anche WAMP[4],
Di fatto i requisiti di piattaforma sono i seguenti:

- **Web server** con supporto a php.
- **Interprete Php** di versione non inferiore alla 4.3.
- **DBMS** relazionale supportato da Php.

[2]Linux, Apache, Postgresql, Php
[3]Linux, Apache, MySQL, Php
[4]Windows, Apache, MySQL, Php

Capitolo 7

Implementazione

7.1 Obiettivi

Si intende realizzare il sistema finale di acquisizione, memorizzazione e presentazione dei dati meteo. Il sistema finale si compone di tre componenti distinte:

- **Database**: memorizza le informazioni, assicurandone la persistenza, e garantendo affidabilità e prestazioni;
- **Crawler**: si occupa dell'acquisizione dei dati e del loro inserimento nella base di dati;
- **Interfaccia Web**: consente l'interazione tra utente e sistema, con funzioni di amministrazione e presentazione.

Le tre componenti comunicano tra di loro nelle modalità riportate nel seguente schema.

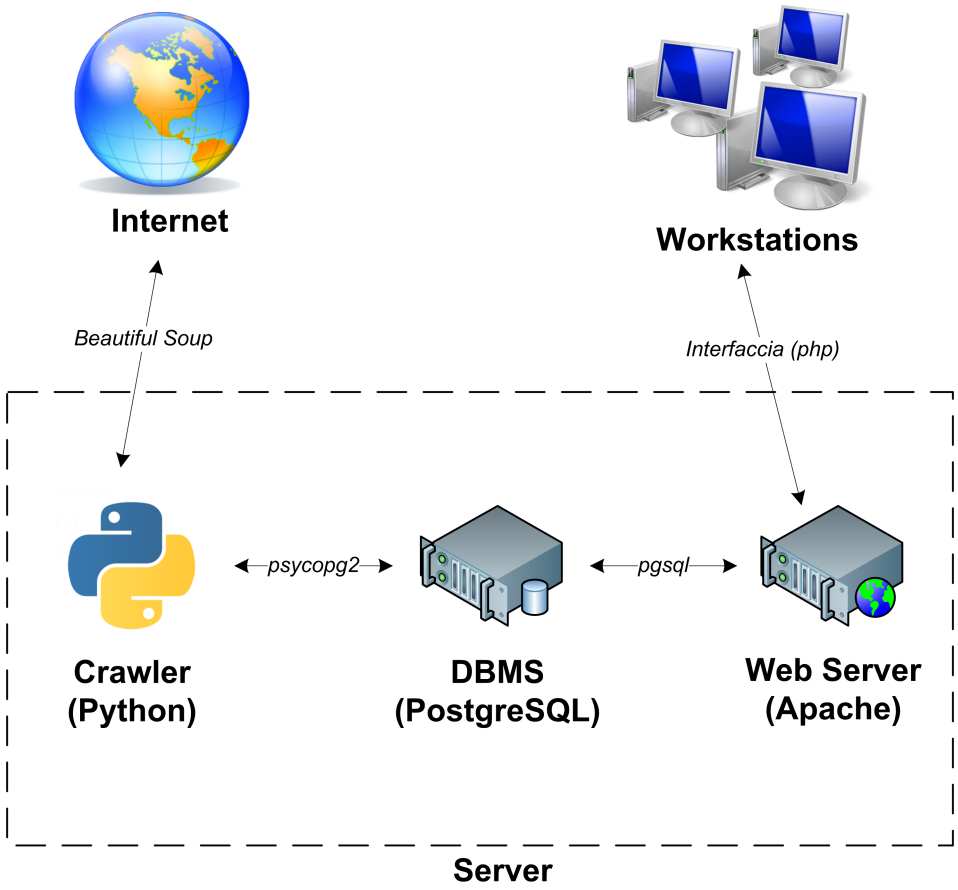


Figura 7.1: Architettura di sistema

Il crawler in prima istanza interroga il DBMS per ottenere gli identificativi delle località che si è scelto di monitorare. Ottenuti tali identificativi, il crawler procede con il recupero e l'estrazione delle informazioni dalla fonte che è stata scelta (*ilmeteo.it*).
Infine il crawler procede con l'inserimento delle informazioni nel database.

L'interfaccia web risponde alle richieste dell'utenza: recupera le informazioni interrogando il DBMS, le visualizza in formato testuale e grafico, e all'occorenza le rielabora.
L'interfaccia web inoltre viene utilizzata dall'utente per modificare l'elenco delle città da monitorare.

7.2 Database: Postgresql

Come DBMS si è scelto di utilizzare *Postresql*.

7.2.1 Progettazione concettuale

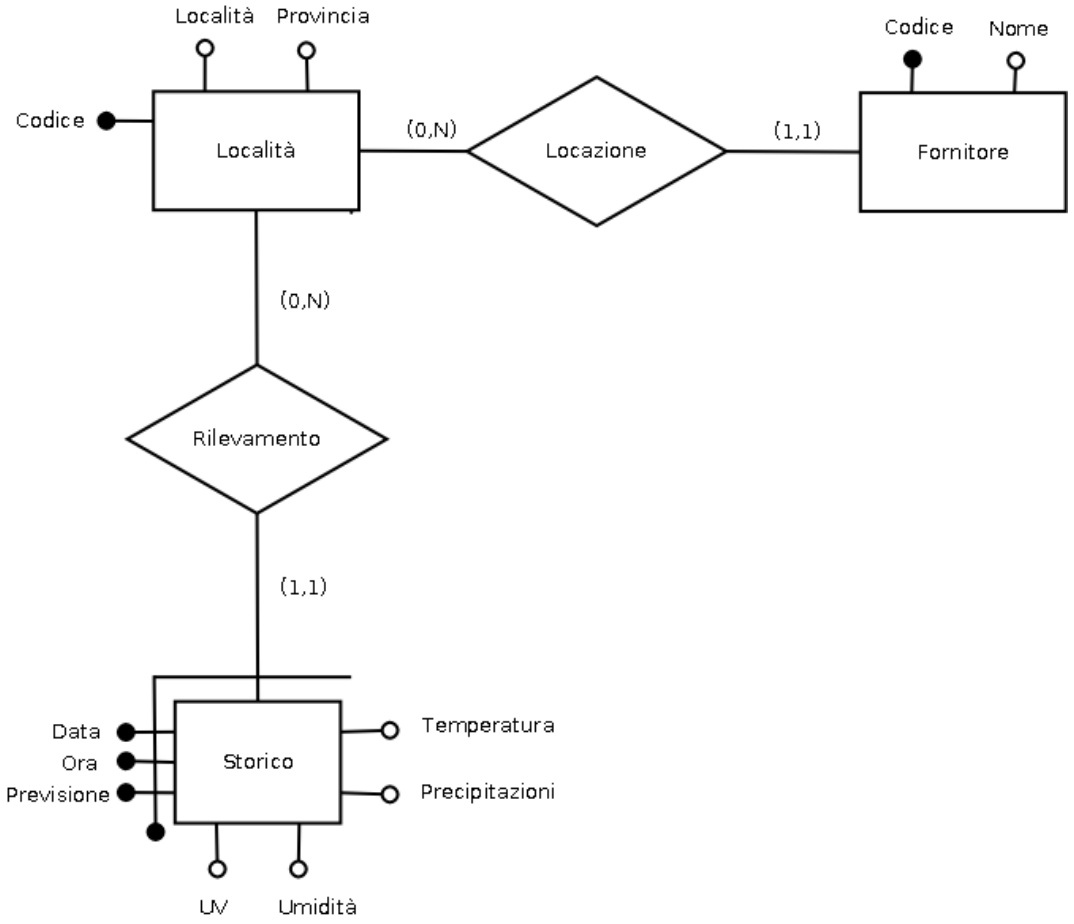


Figura 7.2: Database: schema concettuale (ER)

7.2.2 Dizionario dei dati

Entità

Entità	Descrizione	Attributi	Identificatore
LOCALITÀ	Luogo geografico	Codice, Località, Provincia	Codice
FORNITORE	Produttore	Codice, Nome	Codice
STORICO	Dato meteo	Data, Ora, Previsione, Temperatura, Precipitazioni, Umidità, UV	*Localita*, Data, Ora, Previsione

Associazioni

Associazione	Attributi	Entità collegate
Locazione	–	Fornitore (1,1), Localita (0,N)
Rilevamento	–	Localita (0,N), Storico (1,1)

Regole di vincolo

Per rappresentare fedelmente la realtà di interesse è necessario definire delle regole di vincolo per concetti altrimenti non esprimibili utilizzando il modello E.R.

- (RV1) Gli attributi *Località* e *Provincia* di *Località* non devono essere nulli.
- (RV2) L'attributo *Nome* di *Fornitore* non deve essere nullo.
- (RV3) L'attributo *Codice* di *Località* deve essere positivo.
- (RV4) L'attributo *Codice* di *Fornitore* deve essere positivo.
- (RV5) L'attributo *Data* di *Storico* deve essere una data valida.
- (RV6) L'attributo *Ora* di *Storico* deve essere compreso tra 0 e 23.
- (RV7) L'attributo *Previsione* di *Storico* deve essere compreso tra 0 e 4.

- (RV8) L'attributo *Precipitazioni* di *Storico* deve essere positivo.
- (RV9) L'attributo *Umidità* di *Storico* deve essere positivo.
- (RV10) L'attributo *UV* di *Storico* deve essere positivo.

7.2.3 Traduzione logica: modello relazionale

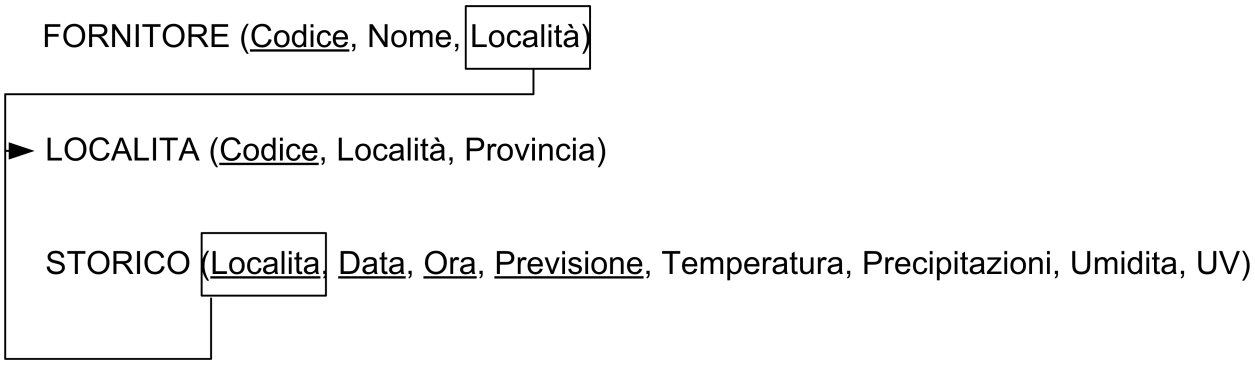


Figura 7.3: Database: schema logico

Regole di vincolo

Si ereditano tutte le regole di vincolo dello schema concettuale.

7.2.4 Codice SQL

```
CREATE TABLE Localita(
    Codice      integer,
    Localita    varchar(200) NOT NULL,
    Provincia   varchar(100) NOT NULL,
    CONSTRAINT "localita_PK" PRIMARY KEY (Codice),
    CONSTRAINT "localita_codice_CK" CHECK (Codice>0)
);
```

```
CREATE TABLE Fornitori(
    Codice      serial,
    Nome        varchar(200) NOT NULL,
    Localita    integer,
    CONSTRAINT "fornitori_PK" PRIMARY KEY (Codice),
    CONSTRAINT "fornitori_localita_FK" FOREIGN KEY (Localita)
        REFERENCES Localita(Codice) ON UPDATE CASCADE ON DELETE
        CASCADE,
```

```
    CONSTRAINT "fornitore_codice_CK" CHECK (Codice>0)
);
```

```
CREATE TABLE Storico(
    Data            date,
    Ora             smallint,
    Localita        integer,
    Previsione      smallint,
    Temperatura     numeric(3,1),
    Umidita         smallint,
    Precipitazioni  numeric(4,1),
    UV              numeric(3,1),
    CONSTRAINT "storico_PK" PRIMARY KEY (Data,Ora,Localita,
        Previsione),
    CONSTRAINT "storico_localita_FK" FOREIGN KEY (Localita)
        REFERENCES Localita(Codice) ON UPDATE CASCADE ON DELETE
        CASCADE,
    CONSTRAINT "storico_ora_CK" CHECK (ora>=0 and ora<=23),
    CONSTRAINT "storico_previsione_CK" CHECK (previsione>=0 and
        previsione<=4),
    CONSTRAINT "storico_precipitazioni_CK" CHECK (precipitazioni
        >=0),
    CONSTRAINT "storico_umidita_CK" CHECK (umidita>=0),
    CONSTRAINT "storico_UV_CK" CHECK (UV>=0)
);
```

7.2.5 Stima del tasso di crescita dei dati

Per disporre di una stima della quantità di spazio occupata dall'archiviazione dei dati, si è calcolato il numero totale di righe, inserite su base annua, per ogni singola località da monitorare.

Descrizione	**Valore**	**Risultato**
Ore in un giorno	24	
Ore in un anno	24 * 365	8.760
Profondità di ogni dato	1 dato reale, 4 previsioni	5
Numero totale di righe	8.760*5	**43.800**

Tabella 7.1: Occupazione annuale in righe per una singola località

Con il sistema a regime si è stimato di monitorare un numero di località pari a 200 circa. Disponendo di tale stima è possibile calcolare il numero complessivo di righe, inserite su base annua, per tutte le località da monitorare.

Descrizione	**Valore**	**Risultato**
Righe per una località	43.800	
Numero di località	200	
Numero complessivo di righe	43.800*200	**8.760.000**

Tabella 7.2: Occupazione annuale in righe per tutte le località

Ora è possibile calcolare (con valore indicativo) la dimensione di una singola riga.

Attributo	**Tipo**	**Dimensione (byte)**
Data	date	4
Ora	smallint	2
Località	integer	4
Previsione	smallint	2
Temperatura	numeric(3,1)	10
Umidità	smallint	2
Precipitazioni	numeric(4,1)	10
UV	numeric(3,1)	10
Totale		**44**

Tabella 7.3: Dimensione di una singola riga

Infine si può calcolare la crescita dello spazio fisico occupato ogni anno dalla base di dati.

Descrizione	**Valore**
Dimensione di una riga	44 Byte
Numero di righe	8.760.000
Totale	**385.440.000 Byte**

Tabella 7.4: Dimensione occupata annualmente

Riassumendo, stimando a 200 il numero di località da monitorate, ogni anno il sistema deve memorizzare circa 368 MB di dati.

7.3 Librerie per l'interfacciamento del crawler con il database

Software	**Licenza**	**OS**	**Versione Python**	**DB API 2.0**	**Nativo (libpq)**
Psycopg	LGPL	Unix, Win32	2.4-2.6	si	si
PyGreSQL	BSD	Unix, Win32	2.3-2.6	si	si
ocpgdb	BSD	Unix	2.3-2.6	si	si
py-postgresql	BSD	qualsiasi	3.0+	si	no
bpgsql	LGPL	qualsiasi	2.3-2.6	si	no
pg8000	BSD	qualsiasi	2.5+/3.0+	si	no

Tabella 7.5: Librerie Python per PostgreSQL

7.3.1 Psycopg

Psycopg è una libreria per l'interfacciamento ad un database PostgreSQL attraverso il linguaggio Python.
I suoi punti di forza sono la capacità di supportare completamente le Python DB API 2.0 e la sicurezza dei suoi thread che possono condividere le connessioni. Venne creato per le applicazioni che usano in maniera pesante il multi-thread, creando e distruggendo molti cursori che compiono numerose operazioni di inserimento (INSERT) o modifica (UPDATE) sul database.

Pyscopg 2 è quasi una completa rivisitazione della prima versione. Le sue caratteristiche completano il protocollo libpq (v3), le funzioni COPY TO/COPY FROM e l'adattamento di tutte le classi base di dati di Python: stinghe (anche in unicode), interi, long, float, buffer (oggetti binari), boleani e i dati di tipo datetime. Inoltre supporta le query in unicode e le liste Python mappate in array PostgerSQL.

7.4 Codice sorgente del crawler

Qui si propone il codice sorgente del crawler relativo alle funzioni di lettura e inserimento dei dati nel database.

```
import psycopg2

db_name = "meteo"
db_user= "postgres"

[ ... ]

conn = psycopg2.connect("dbname=" + db_name + " user=" + db_user
    )
cur = conn.cursor()
cur.execute("SELECT Codice FROM Localita")
rows = cur.fetchall()
for row in rows:

    [ ... ]

    cur.execute("INSERT INTO storico (Data, Ora, Localita,
        Previsione, Temperatura, Umidita, Precipitazioni, UV)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s)",(today, orario,
         location, 0, temperatura, umidita, precipitazioni, UV))
```

```
    conn.commit()

    [ ... ]

cur.close()
conn.close()
```

7.5 Codice sorgente dell'interfaccia web

Qui si propone il codice sorgente dell'interfaccia web relativo alle funzioni di lettura e inserimento dei dati nel database.

```
[ ... ]

<?php
if (isset($_GET["c"])) {
    $codice = $_GET["c"];

    if (isset($_GET["t"])) $tipo = $_GET["t"];
    else $tipo = 't';

    if (isset($_GET["d"]))  $data = $_GET["d"];
    else $data = date("Y-m-d");

    $conn = pg_connect($db_param);
    $result = pg_query($conn, "SELECT * FROM localita WHERE
        Codice=$codice;");
    $row = pg_fetch_row($result);
    $localita = $row[1];
    $provincia = $row[2];
    echo "<h3>$localita</h3>";

    $conn = pg_connect($db_param);
    $result = pg_query($conn, "SELECT DISTINCT Data FROM storico
        WHERE Localita=$codice ORDER by Data;");
    echo "- ";
    while ($row = pg_fetch_row($result)) {
    if ($data==$row[0]) echo "<a href=\"meteo.php?c=$codice&t=
        $tipo&d=$data\"><b>[$data]</b></a> - ";
    else echo "<a href=\"meteo.php?c=$codice&t=$tipo&d=$row
        [0]\">$row[0]</a> - ";
```

```
    }

    echo "<br>";

    echo "- ";

    if ($tipo=="t") echo "<a href=\"meteo.php?c=$codice&t=t&d=
        $data\"><b>[temperature]</b></a> - ";
    else echo "<a href=\"meteo.php?c=$codice&t=t&d=$data\">
        temperature</a> - ";

    if ($tipo=="p") echo "<a href=\"meteo.php?c=$codice&t=p&d=
        $data\"><b>[precipitazioni]</b></a> - ";
    else echo "<a href=\"meteo.php?c=$codice&t=p&d=$data\">
        precipitazioni</a> - ";

    if ($tipo=="u") echo "<a href=\"meteo.php?c=$codice&t=u&d=
        $data\"><b>[umidita]</b></a> - ";
    else echo "<a href=\"meteo.php?c=$codice&t=u&d=$data\">
        umidita</a> - ";

    if ($tipo=="uv") echo "<a href=\"meteo.php?c=$codice&t=uv&d=
        $data\"><b>[irradiamento UV]</b></a>";
    else echo "<a href=\"meteo.php?c=$codice&t=uv&d=$data\">
        irradiamento UV</a>";

    echo " -";

    echo "<br>";
    echo "<br>";
    echo "<img src=\"graph.php?c=$codice&t=$tipo&d=$data\" style
        =\"border: 1px #132F2F solid;\"/>";
}

[ ... ]
```

7.6 Portabilità del sistema

Il sistema si compone di tre strati fondamentali: dbms, crawler e interfaccia web. La portabilità di ciascuno strato è stata già discussa nei capitoli 5 e 6.
Riassumendo, l'intero sistema risulta facilmente portabile su piattaforme Unix, Win e Mac. Il dbms scelto è PostgreSQL, che è disponibili per tutte le principali

piattaforme. Anche l'interprete python è multipiattaforma, così come il web server Apache e l'interprete PHP.
La piattaforma scelta per lo sviluppo può pertanto essere considerata come piattaforma suggerita, ma non vincola assolutamente la scelta della piattaforma da utilizzare in quanto l'intero sistema è composto da elementi portabili.

Capitolo 8

Test

8.1 Obiettivi

Il sistema progettato nei capitoli 5 e 6, e implementato nel capitolo 7, ora viene messo in funzione e le sue funzionalità vengono collaudate.

Il collaudo è stato suddiviso in due fasi:

- **Collaudo del crawler** Il crawler viene eseguito per recuperare i dati relativi ad alcune località di prova. Si desidera verificare sia la correttezza dell'estrazione delle informazioni, sia la correttezza del loro inserimento nel database.
- **Collaudo dell'interfaccia web** Si utilizza l'interfaccia web nelle modalità proprie dell'utente finale. Si intende controllare che tutte le operazioni di visualizzazione, inserimento e modifica della base di dati siano funzionanti.

8.2 Piattaforma di test

Le procedure di collaudo del sistema sono avvenute utilizzando una macchina virtuale[1]. Come software di virtualizzazione è stato utilizzato Virtualbox, sia su sistema Windows che su sistema GNU/Linux.

[1] il termine macchina virtuale (VM) indica un software che crea un ambiente virtuale che emula il comportamento di una macchina fisica ed in cui alcune applicazioni possono essere eseguite come se interagissero con tale macchina

8.2.1 Oracle Virtualbox

VirtualBox è un software di virtualizzazione commerciale proprietario (con una versione ridotta distribuita secondo i termini della GNU General Public License) per architettura x86 che supporta Windows, GNU/Linux e Mac OS X (beta) come sistemi operativi host, ed è in grado di eseguire Windows, GNU/Linux, OS/2 Warp, OpenBSD e FreeBSD come sistemi operativi guest. Nel gennaio 2007 ne è stata rilasciata una versione ridotta.
VirtualBox supporta la soluzione per la virtualizzazione hardware di Intel VT-x ed, in via sperimentale la soluzione di AMD, AMD-V, ma non usa nessuna delle due per impostazione predefinita.
Il 12 febbraio 2008 Sun Microsystem ha acquistato Innotek GmbH, l'azienda tedesca sviluppatrice di VirtualBox.
Il 27 gennaio 2010 è stata perfezionata l'acquisizione di Sun da parte di Oracle Corporation.

8.2.2 Macchina virtuale

Hardware virtualizzato

- CPU generica x86 32 bit
- Memoria RAM 192 MB
- Hard Drive 3 GB
- NIC 1: Intel Pro/1000 MT (Scheda con bridge)
- NIC 2: Intel Pro/1000 MT (Configurato NAT)

La prima scheda di rete consente di instaurare una rete tra sistema host[2] (IP 10.0.1.1) e sistema guest[3] (IP 10.0.1.2), come se le due macchine fossero collegate da un cavo ethernet incrociato.
Con la seconda scheda di rete, Virtualbox fornisce connettività internet tramite un NAT virtuale, comprensivo di un comodo server DHCP che assegna l'indirizzo al sistema guest.

Sistema operativo

Come sistema operativo è stata utilizzata la distribuzione Arch Linux, che fornisce un sistema GNU/Linux completo. Per maggiori informazioni su Arch Linux si

[2] il sistema fisico che ospita una o più macchine virtuali
[3] il sistema virtualizzato

consulti il capitolo 6.
L'installazione di Arch Linux è avvenuta utilizzando l'ultima immagine ISO (2010.05) e i pacchetti sono stati aggiornati all'ultima versione disponibile nei repository.

Software d'ambiente

- **DBMS PostgreSQL** 8.4.4
- **Interprete Python** 2.6.5
- **Web server Apache** 2.2.15
- **Interprete PHP** 5.3.2
- **Librerie PHPGraphLib** 2.30
- **Librerie python-psycopg2** 2.2.1

Software applicativo

Il software applicativo qui riportato è quello utilizzato sul sistema host (un notebook con Arch Linux).

- **Editor di testo: vim** 7.3.3
- **Editor di testo: gedit** 2.30.3
- **Web browser: Chromium** 6.0.472.62
- **Web browser: Firefox** 3.6.10
- **Editor grafico: GIMP** 2.6.10
- **Emulatore di terminale: lxterminal** 0.1.9

8.3 Collaudo del crawler

Il crawler è stato messo in esecuzione con un elenco di 76 località da monitorare. La schedulazione scelta per il test è:

- **Ogni ora:** Si acquisisce il singolo dato orario attuale che rappresenta il dato reale misurato e definitivo di ogni località.

- **Alle ore 19 di ogni giorno:** si acquisicono i dati previsionali di tutte le 24 ore, riferiti ai 4 giorni successivi a quello di campionamento, per ogni località.

All'inizio di ogni ciclo di esecuzione il crawler memorizza il timestamp[4] di partenza. Quando l'acquisizione dei dati termina, viene eseguita la differenza tra il timestamp corrente e quello di partenza. In questo modo viene calcolata la durata dell'operazione di acquisizione. Infine il crawler attende per un tempo, in secondi, pari alla differenza tra 3600 e la durata dell'acquisizione[5].
Questo garantisce che ciascuna acquisizione comincerà a esattamente a un'ora dall'inizio dell'acquisizione precedente.

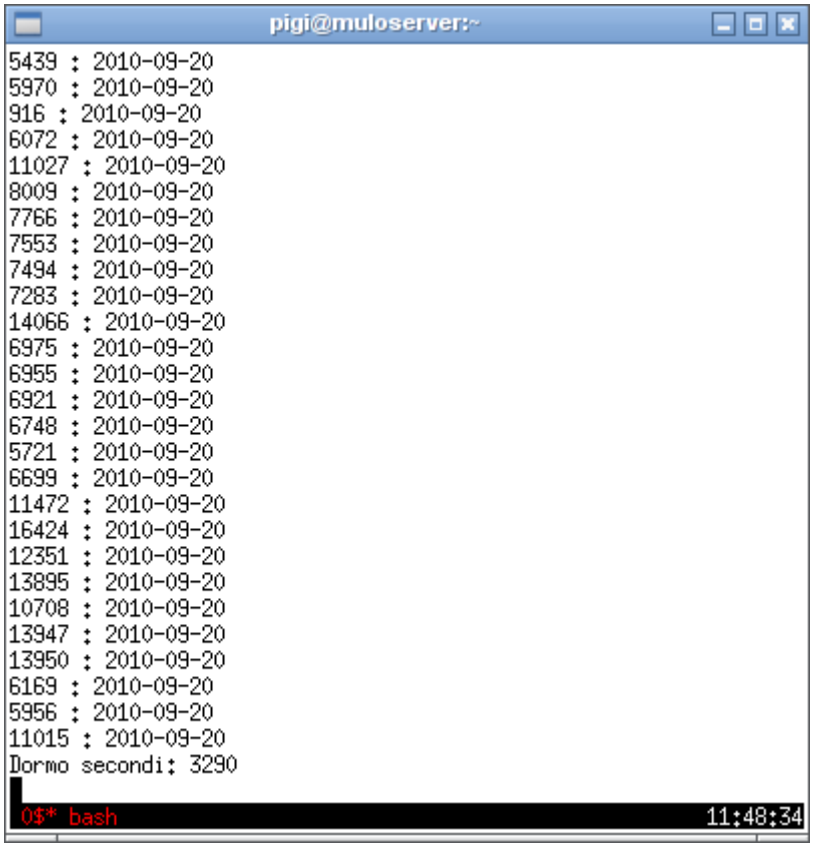
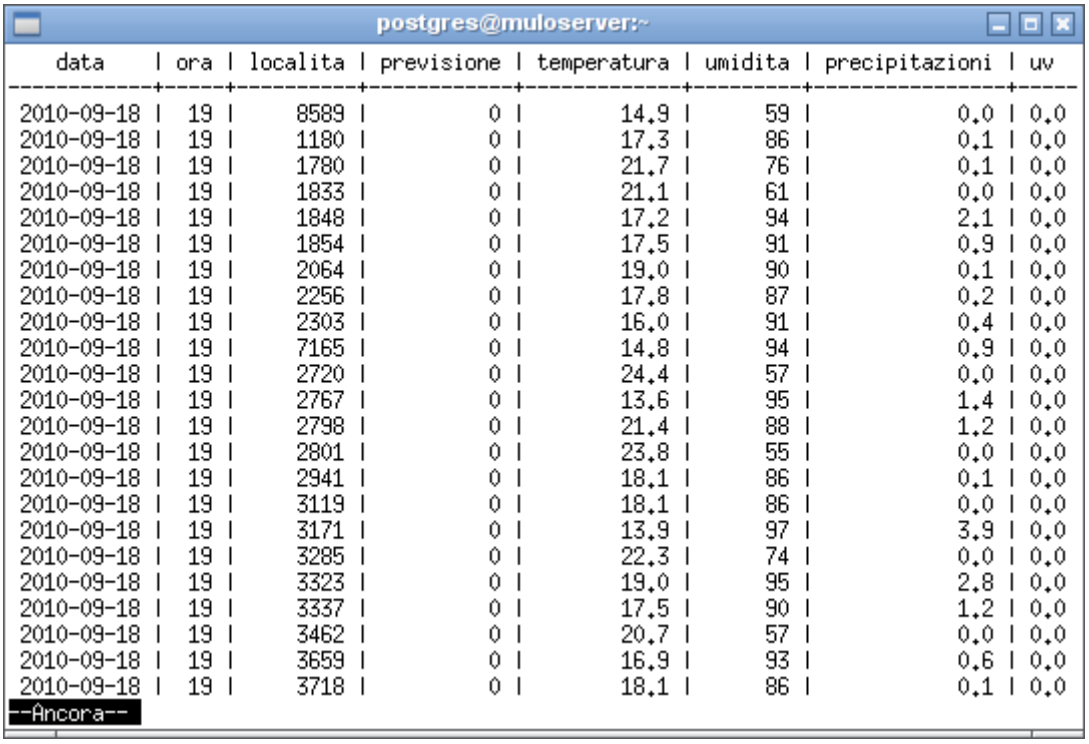


Figura 8.1: Test del crawler: verifica del parsing

[4]una sequenza di caratteri che rappresentano una data e/o un orario per accertare l'effettivo avvenimento di un certo evento. La data è di solito presentata in un formato consistente, in modo che sia facile da comparare con un'altra per stabilirne l'ordine temporale

[5]per i timestamp si utilizza lo unix time. Nei sistemi operativi Unix e Unix-like il tempo viene rappresentato come offset in secondi rispetto alla mezzanotte (UTC) del 1° gennaio 1970

Per verificare la validità dei dati acquisiti si è usato *psql*, il client a linea di comando di PostgreSQL.



data	ora	localita	previsione	temperatura	umidita	precipitazioni	uv
2010-09-18	19	8589	0	14.9	59	0.0	0.0
2010-09-18	19	1180	0	17.3	86	0.1	0.0
2010-09-18	19	1780	0	21.7	76	0.1	0.0
2010-09-18	19	1833	0	21.1	61	0.0	0.0
2010-09-18	19	1848	0	17.2	94	2.1	0.0
2010-09-18	19	1854	0	17.5	91	0.9	0.0
2010-09-18	19	2064	0	19.0	90	0.1	0.0
2010-09-18	19	2256	0	17.8	87	0.2	0.0
2010-09-18	19	2303	0	16.0	91	0.4	0.0
2010-09-18	19	7165	0	14.8	94	0.9	0.0
2010-09-18	19	2720	0	24.4	57	0.0	0.0
2010-09-18	19	2767	0	13.6	95	1.4	0.0
2010-09-18	19	2798	0	21.4	88	1.2	0.0
2010-09-18	19	2801	0	23.8	55	0.0	0.0
2010-09-18	19	2941	0	18.1	86	0.1	0.0
2010-09-18	19	3119	0	18.1	86	0.0	0.0
2010-09-18	19	3171	0	13.9	97	3.9	0.0
2010-09-18	19	3285	0	22.3	74	0.0	0.0
2010-09-18	19	3323	0	19.0	95	2.8	0.0
2010-09-18	19	3337	0	17.5	90	1.2	0.0
2010-09-18	19	3462	0	20.7	57	0.0	0.0
2010-09-18	19	3659	0	16.9	93	0.6	0.0
2010-09-18	19	3718	0	18.1	86	0.1	0.0

Figura 8.2: Test del crawler: verifica dello stato del database con psql

8.3.1 Criticità

Durante la fase di test del crawler sono state riscontrate e gestite le seguenti criticità.

- **Rete** La connettività verso internet non è sempre garantita. Il crawler deve proseguire nell'esecuzione anche quando la rete o la risorsa web non è disponibile.
- **Formato della pagina** Alcune pagine hanno struttura leggermente diversa, a seconda della presenza di dati meteo aggiuntivi riportati dagli utenti di *ilmeteo.it*. Il crawler deve essere robusto nel riconoscere e gestire tale variabilità.

8.4 Collaudo dell'interfaccia web

Si intende controllare che tutte le operazioni di visualizzazione, inserimento e modifica della base di dati siano funzionanti.

Come primo test sono state verificate le funzioni di presentazione.

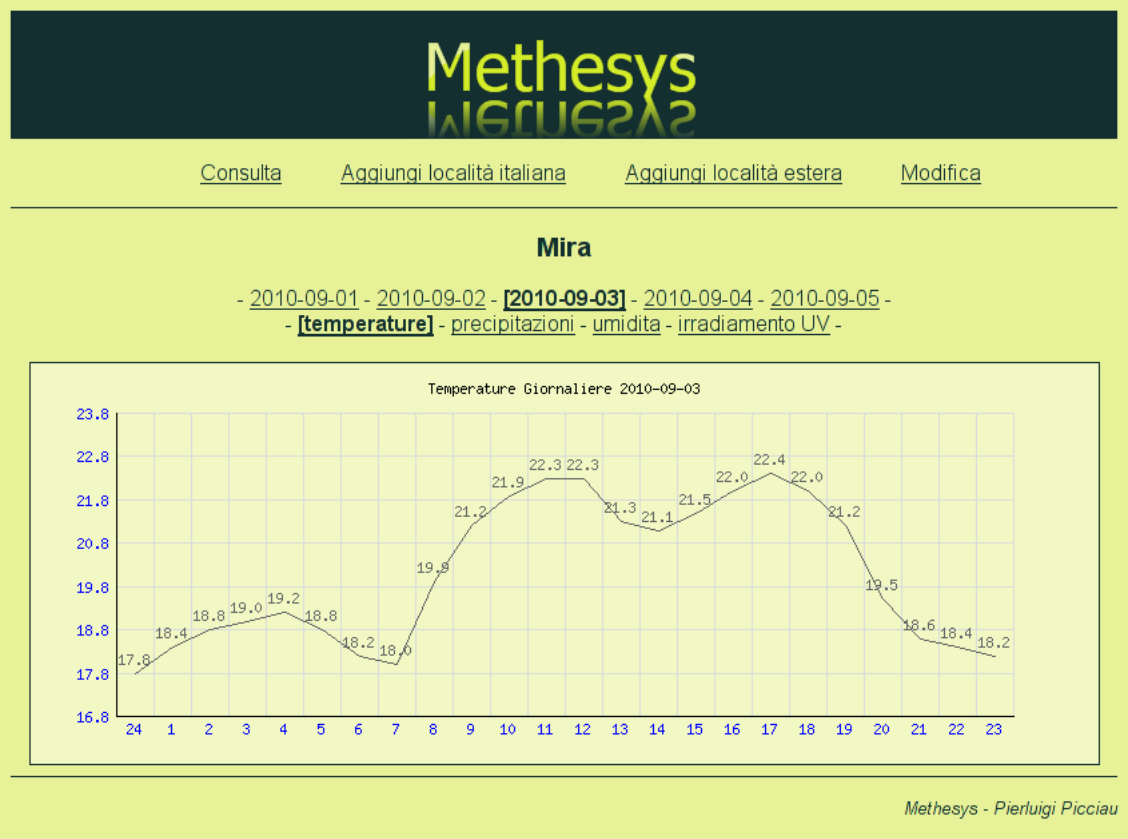


Figura 8.3: Test dell'interfaccia web: grafico delle temperature giornaliere di Mira (VE)

Per una corretta gestione della scala dei grafici è stato necessario calcolare dinamicamente l'intervallo dei dati.

Come secondo test sono state verificate le funzioni di ricerca di una nuova località. In particolare è stata controllata la gestione dei risultati nulli e dei risultati multipli.

Figura 8.4: Test dell'interfaccia web: ricerca di una località

Per una corretta gestione dei risultati multipli è stato necessario modificare lo script php che si occupa del parsing.

Come terzo e ultimo test è verificata la funzione di inserimento di una nuova località.
In particolare è stata controllata la gestione degli accenti e dei caratteri non standard.

Figura 8.5: Test dell'interfaccia web: aggiunta di una località

Per una corretta gestione dei caratteri non standard è stato necessario impostare l'encoding della pagina html a UTF8.

8.4.1 Criticità

- **Codifica dei caratteri** Alcune località hanno nomi contenenti caratteri accentati (località italiane) o con dieresi (località estere). L'interfaccia web deve gestire a pieno la codifica UTF8 per garantire il supporto a un set di caratteri estesi.
- **Gestione delle eccezioni** Alcuni inserimenti possono produrre violazioni di vincolo di chiave primaria. L'interfaccia web deve prevenire queste situazioni.

Bibliografia

[1] AA.VV., *Appunti Universitari per Basi di Dati*

[2] Pilgrim, M., *Dive into Python, versione elettronica,* *http://it.diveintopython.org/*

[3] Giacomini, D., *Appunti di informatica libera, versione elettronica,* *http://a2.pluto.it/*

[4] Sito di PostgreSQL, *http://www.postgresql.org*

[5] Sito di Arch Linux Italia, *http://www.archlinux.it*

[6] Sito di Beautiful Soup, *http://www.crummy.com/software/BeautifulSoup*

[7] Sito di Apache, *http://www.apache.org*

[8] Sito di Php, *http://www.php.net*

[9] Sito di PhpGraphLib, *http://www.ebrueggeman.com/phpgraphlib/*

[10] Sito di Python, *http://www.python.org*