Università degli Studi di
Modena e Reggio Emilia

Facoltà di Ingegneria – Sede di Modena

Corso di Laurea in Ingegneria Informatica – *Nuovo Ordinamento*

Tecnologia database per l'analisi di log file di Web server

Relatore:

Prof. Daniele Montanari

Correlatore:

Prof. Sonia Bergamaschi

Candidato:

Tania Farinella

Anno Accademico 2004-2005

Ai miei genitori

Ringraziamenti

Desidero ringraziare anzitutto il Professor Daniele Montanari per i consigli e per l'aiuto determinante fornitomi in tutto il periodo di ricerca e nella stesura del presente documento. Un ringraziamento particolare va alla Professoressa Sonia Bergamaschi per la disponibilità.

Esprimo tutta la mia gratitudine ai miei genitori per il loro costante supporto economico e psicologico nell'intero arco dei miei studi, per aver assecondato tutte le mie richieste, per tutti i sacrifici che hanno fatto per me. Ringrazio tutti i parenti, in particolare mia zia Luigia.

Ringrazio tutti i professori dell'ITIS L. DaVinci di Carpi per la formazione scolastica che mi ha consentito di superare con facilità buona parte degli esami.

Ringrazio i miei colleghi e compagni di corso grazie ai quali la vita universitaria è stata piacevole. Ringrazio Matte e le nostre chattate (e i pacchi che mi ha tirato! … ma gli voglio bene lo stesso), Mauri, Musso (e i commenti ai prof. durante le lezioni), Robby, Carru, Rivva, l'Elena, Mattia, Elena, Ciccio, Enry, From, Pier e Daniele. Ringrazio Devid per aver sopportato la mia ignoranza informatica e per avermi aiutato per tutto il periodo della tesi.

Infine, ringrazio i miei amici, che mi hanno reso felice in questi anni. Ringrazio l'Angy (e il parco di Migliarina) che mi sopporta sempre e mi aiuta ad affrontare le mie "disgrazie". Ringrazio il mio fratellino acquisito Luigi per il sostegno nelle questioni amorose, Fre, Iatto, Jack, Balla e tutto il gruppo. Ringrazio le mie ciccie, l'Aly, la Lu (che adesso si è persa) e la Liz (… e l'alcool), che mi hanno accompagnato nelle serate di quest'ultimo anno. Ringrazio Pà per i nostri viaggi matematici, Giulio per quelli filosofici, e Umbro per i SUOI viaggi. Ringrazio la Naty per gli anni vissuti insieme.

Grazie a tutti

Tania Farinella

Indice generale

Indice delle figure

Indice delle tabelle

Introduzione

Quando un visitatore accede ad un sito web lascia una traccia del suo passaggio in un archivio locale (*log*) che può registrare l'ora, l'indirizzo IP dell'utente, l'oggetto della richiesta (pagina web, immagine, video), il browser utilizzato ed altri misurabili specificati dai gestori del sito.

Analizzare i dati di un file di log può servire per ottenere statistiche comparabili nel tempo (settimanalmente, mensilmente, trimestralmente) secondo le esigenze di monitoraggio e/o di traffico del sito; correggere e/o modificare il codice HTML e/o intervenire sulla struttura del sito, ottimizzare l'investimento su motori di ricerca e/o su portali analizzando il traffico da questi generato verso il sito; verificare l'andamento di una campagna promozionale del sito confrontando i risultati delle analisi per l'intervallo desiderato (giornaliero, settimanale, ecc.); scoprire utilizzi indesiderati del sito. Mediante l'analisi, infatti, vengono estratte informazioni che indicano quanti visitatori (giorno, ora, area geografica, browser, ecc.) accedono al sito; quanti visitatori provengono da motori di ricerca e portali; quali pagine del sito sono più richieste; quali sono le parole e le frasi con cui il sito è stato trovato nei motori di ricerca[1].

Per ricavare queste ed altre informazioni si può procedere in diverse maniere: si può scegliere di analizzare i file semplicemente leggendoli sequenzialmente senza utilizzare ulteriori strutture dati oltre al log stesso, oppure si può usare l'ausilio di strutture complesse per organizzare i contenuti dei file log e poi ricavarne informazioni la cui estrazione necessita elaborazioni articolate.

Un'altra caratteristica significativa di questi dati è il volume: un sito popolare può produrre decine o centinaia di megabyte di dati log al giorno, quindi un sistema dedicato alla loro analisi deve essere in grado di scalare la propria capacità e performance in modo efficiente al crescere dei volumi trattati.

È stato quindi ideato un progetto software per analizzare i file log provenienti da web server. Il nucleo della trattazione è un database dedicato alla memorizzazione e all'organizzazione di questo tipo di dati, che renda la massima efficienza in termini di possibilità di effettuare analisi, qualità e raffinatezza di queste ultime, complessità delle interrogazioni al database e infine tempi di elaborazione. Il database è stato progettato tenendo conto di due macro-obiettivi:

1. Memorizzare *tutti i dati* presenti nei file di log;
2. Ottenere la massima *flessibilità* rispetto alle interrogazioni.

Il primo obiettivo si è tradotto nel creare una struttura database atta a raccogliere la quasi totalità dei dati presenti nei file, pur se corrotti, e nel correlare ciascun dato a riferimenti indicanti gli

[1] http://www.webric.it/italiano/analisi_logs.html

eventuali errori riconosciuti. I dati non caricati nel database in quanto non compatibili (a causa di eccesso di anomalie) sono comunque raccolti in un file. È dunque possibile ricostruire, in qualsiasi istante, l'intero file di log senza alcuna perdita di informazioni: questo per consentire nel caso, ad esempio, in cui vengano riconosciute le cause che generano dati corrotti, di coinvolgerli nell'analisi, utilizzando altri software o adattando la struttura del database. Se la data di una richiesta, ad esempio, è futura, la linea risulta corrotta: in questo caso è evidente che c'è stato un errore nel registrare la data quindi questa informazione non è utilizzabile. In futuro, però, si potrebbe scoprire che il server ha memorizzato per un certo intervallo di tempo le date delle richieste facendo riferimento all'anno successivo piuttosto che a quello corrente: in tal caso si potrebbero correggere le date errate e si potrebbero effettuare nuovamente le analisi coinvolgendo anche queste ultime.

Per raggiungere il secondo obiettivo, i dati presenti nei log file sono stati suddivisi in più componenti possibili: in particolare sono state separate tutte le informazioni che continuano ad avere un senso anche isolate. In questo modo si riescono a combinare le informazioni in più maniere, sia nei modi già utilizzati, sia in modi che, seppur ora possono non sembrare interessanti, in futuro potranno servire a scopi analitici. In ogni linea, ad esempio, il formato del documento richiesto non viene separato dal nome dello stesso, mentre nel database definito viene riconosciuta l'estensione di ogni file richiesto: in questo modo si riescono ad effettuare analisi per determinare i tipi di file che circolano più frequentemente.

Finita la progettazione sono state implementate interrogazioni al fine di ricostruire i report più comuni forniti dai principali software di analisi di log di web server. Definire queste interrogazioni è stato piuttosto semplice e immediato grazie alla struttura creata. Le query utilizzate risultano semplici dal punto di vista logico e, sfruttando gli indici creati nel database, anche i tempi di esecuzione risultano brevi.

Durante la progettazione sono stati utilizzati file di piccole dimensioni (qualche kilobyte, corrispondenti a poche centinaia di linee) ma significativi, modellati ad hoc, per verificare il funzionamento corretto del sistema anche in casi estremi.

Al termine della definizione di ogni fase del progetto esso è stato testato mediante l'utilizzo di un ulteriore file di log: un file di dimensioni consistenti (circa 85 megabyte) contenente 900000 linee corrispondenti a quasi sette mesi di registrazione di richieste. Le fasi di caricamento e pre-elaborazione hanno i tempi di esecuzione più lunghi (dell'ordine di quindici/venti minuti) mentre le fasi di analisi per generare semplici report impiegano tempi brevi (meno di dieci minuti). Un primo risultato significativo è proprio questo: utilizzando un computer di medie prestazioni (Pentium III, 128 MB di RAM, Windows XP) si ottengono i risultati delle analisi (resoconti generali e specifici) del file descritto in meno di dieci minuti.

In seguito si è passati a effettuare analisi più complesse sia dal punto di vista logico che implementativo, riguardanti sessioni e percorsi, più onerose in termini di tempi di esecuzione (nell'ordine di qualche ora). Per sessione si intendono le pagine del sito viste da ogni persona in un arco di tempo ristretto. Riconoscere persone e pagine è un'operazione complessa e non sempre possibile con certezza totale, dunque chi effettua analisi di questo tipo definisce questi termini indicando le approssimazioni effettuate. È stata scelta una definizione piuttosto semplificata di sessione e anche in questo caso è stato immediato ottenere i risultati voluti; la struttura permetterebbe di lavorare scegliendo definizioni di sessione più complesse.

Le stesse considerazione valgono per i percorsi, che indicano in quale ordine vengono visitate le pagine del sito considerato. È analizzando i percorsi che è stato ottenuto un altro obiettivo significativo. I software più diffusi permettono di determinare quali sono (e con quale frequenza) i percorsi seguiti in ingresso e in uscita da una determinata pagina. Consideriamo, ad esempio, un sito che abbia, tra le altre, una pagina relativa alle informazioni meteo. Un software comune potrebbe determinare che cento utenti hanno visitato la home e poi direttamente la pagina meteo mentre altri duecento sono passati per la pagina relativa ai viaggi e quindi sono arrivati a quella relativa al meteo. Anche col database creato è possibile ricavare tali informazioni, ma è anche possibile determinare i percorsi seguiti prima e dopo la visita consecutiva di *due* pagine del sito. È possibile, cioè, rispondere a domande del tipo "in quanti casi è stata visitata la pagina dei viaggi e immediatamente dopo quella meteo?", "Quali pagine sono state visitate prima di queste due?", "Quali, invece, sono state visitate in seguito alla visualizzazione consecutiva di viaggi e meteo?". Non sarebbe, infine, complesso effettuare le stesse valutazioni relativamente a un numero variabile di pagine visitate consecutivamente piuttosto che solamente due pagine.

Questa tesi è organizzata come segue. Il capitolo1 inquadra il problema dell'analisi dei log e illustra le caratteristiche salienti di alcuni strumenti attualmente in uso. Il capitolo 2 descrive il progetto logico del database e le procedure pre- e post- elaborazione. Il capitolo 3 descrive la realizzazione di alcuni report comunemente prodotti dall'analisi dei log. Il capitolo 4 affronta la classe di problemi avanzati riguardanti i percorsi e le sessioni utente in un sito. Il capitolo 5, infine, raccoglie alcune indicazioni per successivi sviluppi di questo lavoro.

1. Obiettivi, caratteristiche e strumenti dell'analisi dei log

Quando utenti interagiscono con articoli, siti web o con la società in generale, si riescono ad accumulare informazioni, su di loro e sui loro comportamenti, utili poi nel campo decisionale. Le interazioni visitatore-articolo comprendono la cronologia degli acquisti, la cronologia della pubblicità e le informazioni sulle preferenze. La cronologia degli acquisti è un elenco di prodotti e di date di acquisto. La cronologia della pubblicità indica quali articoli sono stati presentati a un visitatore. Le informazioni sul percorso dei clic è una cronologia dei collegamenti ipertestuali selezionati da un visitatore. Le statistiche visitatore-sito sono generalmente tipiche di una sessione e indicano, ad esempio, il tempo totale, le pagine visualizzate, il ricavo e il guadagno per sessione di un visitatore. Le informazioni visitatore-società possono comprendere il numero totale di clienti presentati da un visitatore, il guadagno totale, il numero totale di pagine visitate, il numero di visite al mese, la data dell'ultima visita e possono comprendere valutazioni di marche (elenchi di concetti positivi o negativi che un visitatore associa alla marca stessa, che possono essere misurati considerando periodicamente i visitatori).

Per utilizzare tecniche di estrazione di informazioni su un sito web, occorre riconoscere e memorizzare le caratteristiche dei visitatori, oltre alle loro interazioni, che comprendono informazioni demografiche, psicologiche e tecnologiche. Le informazioni demografiche sono attributi reali quali l'indirizzo, il reddito, la responsabilità di acquisto o la proprietà di apparecchiature per il tempo libero. Le informazioni psicologiche sono i profili psicologici che possono essere rilevati in un'analisi, come il fatto di essere eccessivamente protettivi nei confronti dei figli, la tendenza ad effettuare acquisti impulsivamente, l'interesse per la tecnologia e così via. Le informazioni tecnologiche si riferiscono agli strumenti del visitatore, ad esempio, il sistema operativo, il browser, il dominio e la velocità del modem utilizzato. Se si dispone di un numero telefonico o di un indirizzo spesso è possibile ottenere informazioni di tipo demografico o psicologico tramite provider di servizi di marketing diretto. Le informazioni sugli articoli comprendono le informazioni sui contenuti web (il tipo di media, la categoria del contenuto, l'URL) come pure informazioni su prodotti (codice, categoria, colore, dimensione, prezzo, margine di guadagno, quantità disponibile, livello di promozione)[2].

[2] Tratto da http://www.e-conomy.it/Risorse/e-intelligence/data_mining.htm

1.1. Obiettivi dell'analisi dei log

1.1.1. Web Marketing

Il grande vantaggio del marketing sul web è rappresentato dal fatto che è possibile misurare le interazioni dei visitatori in modo più efficace rispetto ai negozi di vecchio stampo o rispetto al direct mailing (invio di mail pubblicitarie ai clienti registrati alla mailing list). Grazie ai sistemi di analisi possono essere presi in considerazione obiettivi del tipo:

- Aumentare la media di pagine visualizzate per sessione;
- Aumentare il guadagno medio per visita;
- Ridurre la quantità di prodotti restituiti;
- Aumentare il numero di clienti;
- Aumentare la conoscenza del prodotto;
- Aumentare la velocità di ritorno (per esempio il numero di visitatori che sono ritornati entro trenta giorni);
- Ridurre il numero di clic per uscire (numero medio di pagine visualizzate per concludere un acquisto o per ottenere le informazioni desiderate);
- Aumentare la velocità di conversione (risultati per visita);

Se il sito è attrezzato per memorizzare le caratteristiche dei visitatori, dei contenuti e di interazione ed è stata identificata una serie di obiettivi di marketing misurabili, ci si trova in una situazione migliore rispetto alla media. Quello che bisogna fare è sfruttare le informazioni contenute nei dati. Grazie alle informazione ricavate dall'analisi, gli esperti di marketing effettuano processi di targeting e personalizzazione. Possono, ad esempio, selezionare le persone che ricevono un annuncio pubblicitario fisso per migliorare il profitto, il riconoscimento di un prodotto o altri risultati misurabili. I siti web con visitatori di un certo livello generalmente addebitano quote superiori per gli spazi pubblicitari. Sui siti in cui i visitatori si registrano, la pubblicità può essere indirizzata in base alle informazioni demografiche. Ad esempio, persone che vivono in parti diverse del paese o che visitano diversi siti web possono avere diverse propensioni agli acquisti di abbigliamento sportivo, di viaggi o di accessori per auto. Di conseguenza, se ci si rivolge alle persone che più probabilmente acquisteranno il prodotto proposto, sarà possibile ridurre i costi per le campagne pubblicitarie e migliorare il profitto totale. Alcuni siti permettono di destinare gli annunci pubblicitari in base alla teoria che le informazioni della registrazione DNS forniscono l'ubicazione fisica dell'indirizzo IP. Tuttavia, poiché gli ISP spesso condividono un pool di indirizzi IP, questo non è un metodo affidabile. Le tecniche di analisi permettono di selezionare i criteri di indirizzamento di una campagna

pubblicitaria e di identificare la combinazione di criteri che ottimizza il profitto. La personalizzazione consente di selezionare gli annunci pubblicitari da inviare a una determinata persona, incrementando i ricavi poiché il cliente vede più articoli di possibile interesse. Si può cercare, inoltre, di identificare gli articoli che verranno probabilmente acquistati o visti nella stessa sessione. Se si inseriscono dei riferimenti a tali articoli sulla stessa pagina in un catalogo web, si potrebbe indurre il visitatore ad acquistare o visualizzare qualche altro prodotto dimenticato. Se si definisce una promozione su un prodotto in un gruppo omogeneo, si potranno probabilmente incrementare gli acquisti di altri prodotti dello stesso gruppo. In situazioni in cui si dispone di pagine statiche di catalogo ci si basa sul visitatore per selezionare la prima pagina del catalogo da visualizzare e quindi si presentano gli articoli associati come vendite incrociate. Infine è possibile effettuare stime e previsioni: si cerca di indovinare un valore sconosciuto quando si conoscono altre informazioni su una persona, ad esempio la probabilità di acquistare un'automobile entro l'anno successivo, quando una persona non l'ha ancora fatto, oppure il numero previsto di azioni che verranno acquistate da una persona nel corso dell'anno successivo. Se non si conosce il reddito di una persona, un sistema per la stima potrebbe identificare altre variabili che sono correlate con il reddito, ad esempio il luogo in cui vive, le preferenze a livello automobilistico, la posizione lavorativa, e trovare altre persone con caratteristiche analoghe utilizzandole per stimare il reddito e un valore di confidenza. La previsione permette di calcolare importanti attributi futuri di una persona, ad esempio, il valore attribuito al denaro, l'intervallo della visita successiva, la velocità di apprendimento, l'interesse alle promozioni e così via, basandosi sullo stesso approccio. Questi valori possono essere utilizzati in applicazioni di personalizzazione. Chi si occupa di marketing spesso aggrega le informazioni per comprendere il comportamento di gruppi di clienti. Anche l'aggregazione o la valutazione di eventi passati in base a diverse dimensioni, ad esempio la categoria dei visitatori, la categoria del contenuto, il referente e la data, possono fornire utili informazioni. Ciò può aiutare gli esperti di marketing a manipolare i dati per scoprire quali attributi dei prodotti o caratteristiche del sito possono risultare interessanti al maggior numero di clienti[3].

[3] Tratto da http://www.e-conomy.it/Risorse/e-intelligence/data_mining.htm

1.1.2. Analisi

Un server web è un programma installato su un computer che lavora in rete e attende connessioni dal mondo esterno per fornire documenti su richiesta dei browser. Per comunicare, il server e il browser usano il protocollo di comunicazione asincrona HTTP (*HyperText Transfer Protocol*).

Quando un utente accende il browser e digita un URL il browser si collega all'host cercato e richiede un documento specifico. Il server web si occupa di soddisfare la richiesta e spedisce indietro una risposta: se questo documento esiste, il server lo invia; se invece esso non esiste o se l'accesso non è permesso, allora il server spedisce indietro un messaggio di errore. Il documento inviato come risposta alla richiesta del browser può includere richieste di altri oggetti. Questi sono semplici URL che puntano ad altre risorse, che possono essere un documento, un'immagine, un'applet, uno stream video/audio, o qualsiasi altro oggetto HTML indicabile con un indirizzo. Il browser allora richiede tutti gli oggetti inclusi della corrente pagina dal server nello stesso modo in cui ha richiesto il primo documento, prima di mostrare il contenuto di quella pagina. Questo metodo di comunicazione è chiamato asincrono, perché il browser spedisce molte richieste per i documenti inclusi, senza attendere una risposta dal server prima di spedire la nuova richiesta, usando diversi canali di comunicazione. Poiché le richieste del browser sono spesso gestite da differenti processi del server o da differenti flussi di un processo server, non c'è assolutamente relazione tra le linee del log file (il file che tiene traccia degli eventi), causate dalle risposte dal server a seguito della richiesta di un documento, e dei suoi oggetti inclusi. Per esempio, l'ordine con cui il server registra la trasmissione, avvenuta con successo, del documento stesso e delle immagini in esso contenute, non è prevedibile e dipende dal tipo di documento, oggetto, dalla velocità del server, dal sistema operativo, dalla rete e da molti altri parametri.

Prima di analizzare effettivamente un file di log, però, occorre definire con precisione gli obiettivi di tale processo: questa è la fase più critica, infatti tutta la metodologia successiva è organizzata a seconda di quanto stabilito in questa fase. Dopodiché si passa all'organizzazione dei dati: in tale fase si raccolgono e selezionano i dati necessari per l'analisi e se ne effettua una operazione di pulizia preliminare per individuare variabili non utilizzabili, cioè non adatte all'analisi, o variabili mancanti o errate. I dati così ottenuti vengono organizzati e immagazzinati in banche dati (database). In seguito si specificano i metodi statistici, cioè si scelgono i metodi da utilizzare: tutto ciò dipende fortemente dal problema in esame, dagli obiettivi e dal tipo di dati disponibili. Pertanto i metodi utilizzati possono essere classificati in base allo scopo immediato che si vuole raggiungere. Esistono, essenzialmente, quattro grandi classi metodologiche:

- *Metodi esplorativi*: metodologie interattive, solitamente visuali, che hanno lo scopo di trarre le prime conclusioni dalla massa di dati;
- *Metodi descrittivi*: riguardano sia la classificazione delle osservazioni in gruppi non noti a priori, sia la sintesi delle variabili, che vengono relazionate fra loro, secondo legami non noti a priori;
- *Metodi previsivi*: l'obiettivo è spiegare una o più variabili in funzione di tutte le altre, ricercando, in base ai dati disponibili, delle regole che permettono di prevedere o classificare risultati futuri di una o più variabili obiettivo in funzione di quanto accade alle variabili input;
- *Metodi locali*: in questo caso l'obiettivo dell'analisi è l'individuazione di caratteristiche particolari, relative a sottoinsiemi di interesse del database e non l'analisi globale.

Si procede poi con l'elaborazione dei dati, fase in cui i dati vengono processati attraverso un codice. Si confrontano, quindi, i risultati ottenuti con i diversi metodi e si sceglie il modello migliore di analisi. Nella valutazione della performance di uno specifico metodo concorrono vincoli economici, sia in termini di risorse di tempo, che in termini di qualità e disponibilità dei dati. Infine, una volta verificata la consistenza dei modelli adottati per l'analisi dei dati, li si applica al campo di interesse.

1.1.3. Tecniche utilizzate

Per scoprire le relazioni nascoste tra i dati e costruire di conseguenza dei modelli che le rappresentano, esistono due principali tecniche di analisi o approcci: quelle di verifica e quelle di indagine o rispettivamente approccio *top-down* e approccio *bottom-up*. Spesso ci si riferisce a queste tecniche direttamente con il termine *modelli*, in tal caso si dirà rispettivamente *modelli di verifica* e *modelli di scoperta*.

Nell'approccio *top-down*, utilizzando la teoria statistica, si cerca, durante l'esplorazione, di trovare conferme a fatti ipotizzati o che già si conoscono (ad esempio quali fattori hanno prodotto un risultato conosciuto), o di ampliare la conoscenza su nuovi aspetti di un fenomeno che già si conosce in parte. A questo scopo si utilizzano le tecniche statistiche di clustering, l'analisi fattoriale, i metodi previsionali.

Nell'approccio *bottom-up* si ricercano informazioni utili che si ignorano studiando dati e collegamenti tra loro in modo non aprioristico per costruire ipotesi, ad esempio quali fattori sono le cause più probabili che producono un certo risultato. In questo caso vengono utilizzati strumenti quali la tecnologia delle reti neurali e degli alberi decisionali. Spesso, infatti, affidando la ricerca direttamente ai tool si possono scoprire nuove relazioni e segmentazioni fra i dati, altrimenti nemmeno immaginabili. Una volta individuato un fatto nuovo deve essere verificato con una tecnica di verifica precedente, dato che non è escluso che si possano fare scoperte non valide.

Questo progetto si propone di definire una struttura per la raccolta dei dati che permetta di analizzarli utilizzando sia approcci bottom-up sia top-down. Per il momento sono state implementate alcune semplici analisi esplorative: interrogazioni, più o meno complesse, al database. Sarà possibile, però, utilizzare il database creato per sfruttare le altre tecnologie sviluppate per l'analisi dei dati.

1.2. Analisi di un file log

I file log di web server registrano i valori di alcuni parametri relativi al ciascuna richiesta ricevuta dal server stesso. I dati sono raccolti in formato testuale: quando un utente accede ad una pagina web, il browser inoltra una richiesta di risorse al server, il sistema di allocazione e gestione del sito web da cui proviene la pagina richiamata. Le risorse richieste possono consistere in file web (.html, .php, .asp, ecc.), file immagine o grafici, file audio, file video, nonché particolari applicazioni (ad esempio in Java). Il server web accede alle risorse e le invia al browser dell'utente, che così può visualizzarle sullo schermo. Questa attività di scambio fra browser e server web viene registrata all'interno del log file, e si crea così una sorta di "cronologia" delle richieste rivolte al server dai browser dei vari utenti e delle risposte conseguenti.

1.2.1. Common Log File Format

Le informazioni contenute nel log file normalmente sono registrate nel formato noto come *Common Log File Format*[4]. In questo formato il log file si presenta come un file di testo, (ASCII) in cui ogni richiesta effettuata dal browser al server web corrisponde ad una stringa (hit). Nel log file non sono registrate le sole pagine web richieste, ma anche le risorse ad esse associate, come file audio, file grafici, ecc...Ciascuna risposta del server - che indichi successo, errore, persino timeout (cioè nessuna risposta) - viene registrata dal log file del server. Una tipica linea di un log file nel *Common Log File Format* appare così:

```
82.68.58.90 - al [01/Feb/1998:10:10:00 +0100] "GET /ind.htm HTTP/1.0" 200 48
```

[4] http://www.bacuslabs.com/WsvlCLF.html

La linea è costituita da campi separati da spazi bianchi e definiti come nel seguente schema:

Nome	Valore	Descrizione
remotehost	`82.68.58.90`	Contiene il nome di dominio completo (FQDN ovvero *Full Qualified Domain Name*) del sito che accede al server, o l'indirizzo IP se non si riesce a rintracciare l'hostname DNS. Si tratta, dunque, di una serie di caratteri separati da *punti*. Nel caso di indirizzo IP si hanno quattro stringhe di caratteri numerici, ciascuna rappresentativa di un numero tra 0 e 255 e un totale di tre *punti* ('.') separatori.
Rfc931	-	È un'autenticazione del server: se esiste, il server fornisce un'autenticazione del client a livello TCP; tipicamente, però, questo dato è sostituito dal carattere *meno* ('-') in quanto non viene fornito[5].
auth_user	`alex`	Indica lo username col quale l'utente si è autenticato; nei casi in cui l'accesso anonimo al web è consentito questo campo rimarrebbe vuoto, in tal caso il carattere *meno* viene inserito. Ha invece un valore significativo nel caso di pagine web protette che richiedono una password per l'accesso.

[5] Un Request for Comments (RFC) è un documento che riporta informazioni o specifiche messe a disposizione della comunità Internet. Gli RFC furono inizialmente pubblicati nel 1969 come parte del progetto ARPANET. Il loro manutentore, responsabile della loro edizione e del loro ciclo di vita, è RFC Editor.

[data]	`[01/Feb/1998:10:10:00 +0100]`	Rappresenta la data e il fuso orario relativi alla richiesta. Essendo costituita da due valori (separati da spazi bianchi) questo campo è circondato da parentesi quadre. Nella data sono indicati il giorno, il mese e l'anno della richiesta, separati fra loro dal simbolo *slash* ('/'), seguiti, dopo i *due punti* (':') dall'orario contenente ora, minuti e secondi separati dai *due punti*. Il giorno, l'ora, i minuti e i secondi sono rappresentati da due cifre, il mese da una sigla di tre caratteri e l'anno da quattro cifre. Il fuso orario mostra la differenza oraria tra il luogo in cui si riceve la richiesta e l'ora di Greenwich, ed è rappresentato dal simbolo *più* ('+') o *meno* ('-') seguito da tre cifre.
"richiesta"	`"GET /index.html HTTP/1.0"`	È l'identificazione dell'URL richiesto dal client. Anche questo campo contiene più valori, dunque è circondato da *doppie virgolette*. Contiene il metodo di richiesta o request method (GET per esempio), il nome del documento richiesto (URI), e la specificazione del protocollo (HTTP/1.0 o HTTP/1.1), separati da uno *spazio bianco*.
stato	`200`	È il codice (tre caratteri numerici a formare un numero tra 100 e 699) di classificazione del risultato, identificato dal protocollo HTTP inviato dal server in risposta al client. Indica se il file richiesto è stato rintracciato (si veda il paragrafo successivo).

byte	4839	È il numero di byte della risposta. Alcuni server registrano il numero di byte trasferiti effettivamente, mentre altri server registrano la dimensione del documento; la differenza si ha se l'utente interrompe il trasferimento prima che il documento possa essere trasmesso completamente: in tal caso i byte trasferiti saranno solo una parte di quelli che costituiscono l'intero documento.

Tabella 1 – *Common Log File Format*

1.2.1.1. Codici di risposta http

Di seguito è riportato un sommario del significato dei possibili valori assumibili dal codice di risposta http[6].

1xx: Informazioni Questi codici di risposta indicano un codice provvisorio che dovrebbe essere seguito da un altro.		
100	Continua (Continue).	Il server non ha respinto la parte iniziale della richiesta e il client dovrebbe continuare a inviare il resto della richiesta.
101	Cambio Protocollo (Switching Protocols).	Il server concorda con la richiesta del client di cambiare protocollo.

[6] Per maggiori dettagli vedi http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

2xx Successo Questi codici di risposta indicano che la richiesta del client è stata ricevuta, compresa ed eseguita con successo.		
200	Ok (Ok).	La richiesta è stata eseguita con successo.
201	Creata (Created).	La richiesta è stata eseguita e il risultato è stato inserito in una nuova risorsa che è stata creata.
202	Accettata (Accepted).	La richiesta è stata accettata per effettuare il processo, ma questo non è stato completato.
203	Informazione Da Fonte Non Autorevole (Non-Authoritative Information).	Le intestazioni inviate in risposta non sono definitive, ma sono fornite da una copia locale o da una copia della copia.
204	Nessun Contenuto (No Content).	Il server ha eseguito la richiesta ma non c'è nessun output da inviare in risposta.
205	Contenuto Resettato (Reset Content).	Il server ha eseguito la richiesta e il client dovrebbe ricaricare la visualizzazione del documento.
206	Contenuto Parziale (Partial Content).	Il server ha eseguito la richiesta parziale GET per la risorsa.

3xx: Invio ad altro URI Questi codici di risposta indicano che la richiesta del client è stato inoltrata ad un altro URI, o che quella particolare azione deve essere ricevuta dallo user agent perché esegua la richiesta.		
300	Scelte Multiple (Multiple Choices).	Ci sono diverse risorse che coincidono con la richiesta che quindi è stata inoltrata ad una di esse.
301	Spostato Permanentemente (Moved Permanently).	La risorsa richiesta è stata assegnata ad un nuovo URI permanente ed eventuali richieste future per questa risorsa dovrebbero usare il nuovo indirizzo.
302	Trovato (Found).	La risorsa richiesta risiede temporaneamente sotto un diverso URI. Visto che esso potrebbe venir modificato, continuare ad usare il vecchio URI.
303	Vedi Altro (See Other).	La risposta alla richiesta può essere trovata sotto un diverso URI e dovrebbe essere recuperata da quest'ultimo.
304	Non Modificato(Not Modified).	Il documento non è stato modificato dall'ultima richiesta e dovrebbe essere utilizzata la copia locale nella cache.
305	Usa Proxy (Use Proxy).	Bisogna accedere alla risorsa richiesta attraverso un proxy.
306	(Inutilizzato) (Unused).	Usato in una versione precedente del protocollo.
307	URI Differente Temporaneamente (Temporary Redirect).	La risorsa richiesta risiede temporaneamente sotto un URI differente. Dato che potrebbe essere modificato, continua ad usare l'URI attuale.

4xx: Errore client Questi codici di risposta sono utilizzati nei casi in cui sembra che il client abbia sbagliato.		
400	Richiesta Posta Male (Bad Request).	La richiesta non può essere compresa dal server a causa della sintassi errata.
401	Non Autorizzato (Unauthorized).	La richiesta necessita un'autenticazione dell'utente.
402	Richiesta Pagamento (Payment Required).	Questo codice è riservato a usi futuri.
403	Vietato (Forbidden).	Il server ha compreso la richiesta, ma si rifiuta di eseguirla. Solitamente a causa di permessi di accesso a file sul server.
404	Non Trovato (Not Found).	Il server non ha trovato niente corrispondente all'URI..
405	Metodo Non Autorizzato (Method Not Allowed).	Il metodo richiesto non è autorizzato. Tipicamente quando si cerca di eseguire un normale documento o visualizzare uno script.
406	Non Accettabile (Not Acceptable).	Per quanto riguarda la richiesta del client, non è in grado di occuparsi della risposta.
407	Richiesta Autenticazione Del Proxy (Proxy Authentication Required).	Questo codice è simile al *401* (Non autorizzato), ma indica che il client deve prima autenticarsi col proxy.
408	Timeout Della Richiesta (Request Timeout).	Il client non produce la richiesta entro il tempo in cui il server si era preparato per riceverla.
409	Conflitto (Conflict).	La richiesta non può essere completata a causa di un conflitto col corrente stato della risorsa. Tipicamente si vede sono con client che possono manipolare file sul server remoto.
410	Perso (Gone).	La risorsa richiesta non è più rintracciabile dal server e non si conosce nessun indirizzo da seguire.

411	Estensione Richiesta (Length Required).	Il server rifiuta di accettare la richiesta senza un *Content-Length header* definito.
412	Precondizione non soddisfatta (Precondition Failed).	La congettura data in uno o più oggetti dell'*header* è fallita quando è stata testata sul server.
413	Entità Richiesta Troppo Grande (Request Entity Too Large).	Il server sta rifiutando di eseguire una richiesta perché l'entità di tale richiesta è più grande di quanto il server sarebbe in grado di eseguire.
414	URI Richiesto Troppo Lungo (Request-URI Too Long).	Il server sta rifiutando di servire la richiesta perché l'URI richiesto è troppo lungo.
415	Tipo Di Media Non Supportato (Unsupported Media Type).	Il server sta rifiutando di servire la richiesta perché il client non supporta il formato della risposta.
416	Dominio Richiesto Non Accettabile (Requested Range Not Satisfiable).	Il server non riesce a trovare il dominio specificato nel *range header.*
417	Aspettativa Fallita (Expectation Failed).	Il server non riesce a soddisfare l'aspettativa data nell'*Expect header.*

5xx: Errore Server Questi codici di risposta indicano casi in cui il server si è accorto di aver sbagliato o di non essere in grado di eseguire la richiesta.		
500	Errore Interno Del Server (Internal Server Error).	Il server ha riscontrato una condizione inaspettata che gli impedisce di eseguire la richiesta .
501	Non Implementato (Not Implemented).	Il server non supporta le funzionalità necessarie per eseguire la richiesta.
502	Gateway Errato (Bad Gateway).	Il server è un gateway o un proxy e ha ricevuto una risposta non valida dal server "padre" al quale ha effettuato un accesso per poter eseguire la richiesta.
503	Servizio Non Disponibile (Service Unavailable).	Il server al momento non è in grado di eseguire la richiesta a causa di un temporaneo sovraccarico dello stesso o a causa di operazioni di manutenzione.
504	Timeout Del Gateway (Gateway Timeout).	Il server è un gateway o un proxy e non riceve una risposta entro il tempo a disposizione dal server "padre".
505	Versione HTTP Non Supportata (HTTP Version Not Supported).	Il server non supporta, o si rifiuta di supportare, la versione del protocollo HTTP utilizzata nella richiesta.
6xx: Codice non standard Questi codici di risposta non sono stati standardizzati. Talvolta sono definiti dai programmatori e il loro valore ha significato nel solo ambito di alcuni applicativi.		

Tabella 2 – Codici HTTP

1.2.2. Extended Common Log Format e altri formati

Il formato *Extended Common Log Format*[7] è, come dice il nome, estendibile e permette di registrare un più ampio range di parametri. Il *Common Log File Format*, invece, supportato dalla maggior parte degli strumenti di analisi, non registra dati relativi alle transazioni dei server; è dunque conveniente utilizzarlo in alcuni casi, come in quello di siti sensibili alla pubblicazione di dati personali, in cui è preferibile evitare la registrazione di certi dati; nei casi in cui è desiderabile memorizzare più informazioni vengono invece utilizzati formati più articolati come il formato *Extended Common Log Format*. Questo permette di rendere i log file leggibili da generici strumenti di analisi, infatti all'inizio del log viene trascritta un'intestazione che specifica il tipo di dati registrati. Ogni linea del file può contenere una *directive* o una *entry*.

Una *entry* è una sequenza di oggetti relativi ad una singola transazione HTTP. I campi sono separati da spazi bianchi. Se un campo non è utilizzato il suo valore, anche in questo caso, viene rappresentato dal simbolo *meno*. Una *directive* registra, invece, informazioni riguardanti il processo di logging stesso, e si distingue in quanto è preceduta dal simbolo *cancelletto* ('#').
Di seguito viene mostrato un esempio di file in formato Extended.

```
#Version: 1.0

#Date: 12-Jan-1996 00:00:00

#Fields: time cs-method cs-uri

00:34:23 GET /foo/bar.html

12:21:16 GET /foo/bar.html

12:45:52 GET /foo/bar.html

12:57:34 GET /foo/bar.html
```

La direttiva *Fields* permette di specificare quali campi sono presenti nelle *entry*.

[7] http://www.w3.org/TR/WD-logfile.html

Le linee di un file in formato *Common* possono essere viste, quindi, come entry di un file in formato *Extended*. Solitamente si utilizza il formato *Extended* aggiungendo alla linea del log file (in formato *Common*) i seguenti due campi in una delle due modalità descritte di seguito.

Nome	Valore	Descrizione
CLF	…	Sono i campi del formato Common Log file Format.
user-agent	`Mozilla/2.0 (X11; IRIX 6.3; IP22)`	Indica il software che il client dice di utilizzare (il browser). Anche in questo caso se non è determinato si ha il carattere *meno*.
referrer	`http://foo/bar.html`	è l'URL della pagina contenente un link al documento richiesto nel caso in cui la richiesta per quel documento sia stata generata seguendo un link. Nel caso non sia determinato, si ha il carattere *meno*.

Tabella 3 – Primo uso di *Extended Log File Format*

In alcune modalità, lo *user-agent* ed il *referrer* sono circondati da *doppie virgolette* che talvolta li rendono ambigui in quanto possono essere riconosciuti come URL errati. Ne consegue che la forma mostrata dovrebbe essere preferita.

1.3. Software di analisi dei log

Chiunque pubblica su Internet un suo sito web è interessato a sapere quanti lo visitano, cosa guardano, da dove vengono e magari perché lo fanno. Ogni server web tiene traccia nei file di log di ogni file servito ai suoi client. L'analisi dei log prodotti può essere automatizzata e analizzata per ottenere statistiche interessanti e comprensibili. Esistono in circolazione una moltitudine di log analyzer, programmi che analizzano i log di server web e producono un output, tipicamente in HTML, contenente tabelle, schemi, grafici e statistiche così da riassumere ed elencare innumerevoli dati sul traffico generato da un sito. Esistono moltissime alternative commerciali e anche diversi approcci all'analisi di quest'ultimo. Alcuni tool analizzano i log dei server web; altri utilizzando delle proprie "sonde" all'interno di pagine web che registrano i dati dei visitatori su un server centrale; altri ancora inseriscono pezzi di codice, per esempio in PHP, che memorizza in un database gli accessi. Determinati strumenti si adattano bene ad elaborare grosse quantità di log anche su sistemi distribuiti, a dispetto di altri più limitati, mentre alcuni processano senza problemi pagine dinamiche e i relativi accessi, a differenza di altri che lavorano bene solo su pagine statiche.

Tra i software più diffusi troviamo *AWStatics*, *LogMiner*, *Urchin , Sawmill, FastStats*, *WebAlyzer*[8], *Analog* e *WebTrends*. *FastStats*[9] elabora i log file mediante moduli *C* ottimizzati, li analizza, memorizza le informazioni ricavate in formato *XML* e vanta una buona velocità; *LogMiner*[10], invece, utilizza un database *PostgreSQL; Urchin*[11], oltre alle funzioni generiche, consente di effettuare confronti dei dati raccolti in giorni, settimane, mesi diversi e valutarne l'andamento. *AWStatics*[12] e *WebTrends* permettono di identificare le entry e le exit pages, cioè indicano, per ogni pagina, gli URL (e le relative frequenze) richiesti precedentemente e successivamente la visita della pagina considerata. Anche *Sawmill*[13] supporta questa funzione e, analogamente a *WebTrends*, registra le statistiche in un database ottimizzato che viene aggiornato ogni qual volta arrivino file log nuovi. Questo consente di creare e visualizzare le statistiche praticamente in tempo reale in quanto esse sono generate direttamente dal database.

[8] http://www.mrunix.net/webalizer/
[9] http://www.mach5.com/
[10] http://logminer.sourceforge.net/
[11] http://www.urchin.com/
[12] http://awstats.sourceforge.net/
[13] http://awstats.sourceforge.net/

1.3.1. Analog e WebTrends

Analog e *WebTrends* sono dunque due tra i software di analisi dei log file di web server più diffusi. In entrambi possiamo riconoscere delle caratteristiche principali.

WebTrends[14] è uno strumento completo per l'analisi statistica web, con tools interattivi di analisi e una visualizzazione dei dati avanzata; si occupa di monitorare gli accessi utenti ad un web server, ma anche di analizzarne in profondità ogni suo aspetto funzionale. Le richieste di sistema necessarie a far girare il prodotto sono abbastanza elevate in generale, ma non spropositate se le si inserisce in un'ottica di monitoraggio di siti e reti di "peso" notevole. La tecnologia utilizzata permette di ottenere il massimo delle informazioni possibili se il prodotto viene utilizzato per monitorare web server *Netscape* o *Microsoft*: vengono infatti utilizzati appositi plug-in che permettono la rilevazione di dati client-side, normalmente non visibili nei normali file di log. L'ambiente di lavoro è basato su una console amministrativa, dalla quale si possono impostare tutti i profili necessari con le loro varie opzioni. Per profilo viene inteso tutto l'insieme di personalizzazioni che si possono definire: oltre alle consuete impostazioni, per esempio, si possono configurare particolari monitor per analizzare lo streaming di dati o l'efficienza di un proxy server. Un'opzione molto importante è quella che permette di configurare il tipo di report da creare, che può essere una pagina HTML pubblicata automaticamente, una mail, oppure un file collocato su un sito FTP. I profili creati possono essere "eseguiti" in qualsiasi momento come se fossero programmi separati, producendo il risultato desiderato: la velocità di creazione dipende, oltre alla potenza specifica della macchina utilizzata, anche da opzioni esterne al programma, come ad esempio la risoluzione degli indirizzi numerici dei client mediante DNS. *WebTrends* permette anche di effettuare uno scheduling per la creazione dei report in modo da automatizzare il più possibile le operazioni di analisi e di reportistica. Tra le varie funzioni ce ne sono alcune che permettono di monitorare l'efficienza dei servizi web e di avvertire, attraverso e-mail, il superamento di soglie prestabilite, nonché la possibilità da parte del programma di riavviare il web server monitorato in caso di crash, sfruttando i servizi di *Microsoft NT*.

Analog[15] è un programma che misura l'utilizzo di un server web analizzandone i log file. È stato progettato per essere molto veloce, è gratuito e può essere utilizzato con la maggior parte dei sistemi operativi. Fornisce statistiche generando semplici report che possono essere resi più carini mediante l'utilizzo combinato con *Report Magic*[16], un altro software free. Non è

[14] http://www.webtrends.com/
[15] http:/www.analog.cx/
[16] http://www.reportmagic.org/

necessaria alcuna installazione, occorre solo che i file log siano in un formato leggibile da *Analog*: possono essere utilizzati tutti i formati standard oppure si può indicare il formato utilizzato. La configurazione, se diversa da quella standard, è effettuata tramite comandi che permettono di specificare il file log da considerare, il formato del file, le richieste da considerare e quelle escludere.

1.3.1.1. Differenze: vantaggi e svantaggi

WebTrends registra i risultati delle sue analisi in un database, *FastTrends*, in modo da velocizzare le successive operazioni che hanno per oggetto lo stesso time-frame. Vengono inoltre utilizzate tecnologie di caching avanzate. *FastTrends* è stato sviluppato specificamente per l'analisi di un massiccio numero di dati web, e permette un utilizzo limitato della memoria RAM. Memorizzare tutti i dati in un database rivela molteplici vantaggi. Innanzi tutto facilita l'analisi di file caricati in momenti diversi: in qualsiasi istante è possibile analizzare tutti i dati complessivamente, o specificare quale singolo file considerare. Potendo effettuare interrogazioni, le operazioni di analisi risultano semplificate, e di conseguenza si realizzano indagini più complesse. In questo modo il tempo impiegato per effettuare la fase di analisi (escludendo la fase di caricamento dei dati) e ottenere i risultati risulta notevolmente ridotto.

Analog, al contrario, analizza i file in modalità stream, senza memorizzarli in alcuna struttura. È possibile creare un file di cache dove raccogliere i dati acquisiti nel corso di un caricamento, ma tali serializzazioni bloccano la risoluzione delle informazioni. I principali svantaggi si hanno nel momento in cui si vogliano analizzare dati derivanti da file log diversi: in questo caso l'utente può creare un nuovo file contente tutti i dati d'interesse, oppure utilizzare funzionalità di *Analog* dedicate a questo aspetto, ma in modo vincolante e limitante.

2. Progetto logico del log database

Avendo individuato le caratteristiche principali dell'analisi dei file log di web server, e dopo aver osservato il funzionamento di alcuni software, si è motivati ad implementare un database per raccogliere tutti i dati. La flessibilità è un aspetto sempre più importante, soprattutto nell'ambito informatico, si è dunque scelto di dedicarvi maggiore attenzione. Per questo motivo si è pensato di creare una struttura solida, e soprattutto efficiente, nella quale caricare i dati, per poi minimizzare gli oneri delle analisi, in termini di tempi e complessità delle interrogazioni. Un database, infatti, oltre a memorizzare i dati veri e propri, contiene anche le informazioni sulle loro rappresentazioni, sulle relazioni che li legano, sui collegamenti con dati esterni, sulla struttura complessiva e tutti gli altri elementi atti a consentire le loro manipolazioni richieste dalle applicazioni prevedibili. Un requisito importante di un buon database consiste nel non duplicare inutilmente le informazioni in essa contenute, salvando i dati in tabelle che possono essere collegate. La funzionalità di un database dipende in modo essenziale dalla sua progettazione: la corretta individuazione degli scopi del database e quindi delle tabelle, da definire attraverso i loro campi e le relazioni che le legano, permette poi una estrazione dei dati più veloce e, in generale, una gestione più efficiente.

L'ambiente scelto per la creazione e l'utilizzo del database è *Microsoft SQL Server 2000*, il database relazionale prodotto da *Microsoft*. Nelle prime versioni era utilizzato per basi dati medio-piccole, ma negli ultimi anni è stato utilizzato anche per la gestione di basi dati di grandi dimensioni (come in questo caso). *Microsoft SQL Server* usa una variante del linguaggio *SQL standard* (lo standard ISO certificato nel 1992) chiamata *T-SQL* (*Transact-SQL*).

2.1. Pre-elaborazione

Per caricare i dati nel database è stato utilizzato il comando *BULK INSERT*. Quando si esegue questa istruzione, i valori contenuti in un file vengono inseriti in una o più righe della tabella o vista indicata. Si può specificare un elenco dei nomi delle colonne in cui si desidera inserire i dati, oppure omettere tale elenco: in tal caso i dati vengono inseriti in tutte le colonne della tabella o vista. Se nell'elenco non sono indicate tutte le colonne di una tabella o vista, nelle colonne non specificate viene inserito il valore *NULL* o l'eventuale valore predefinito associato a ogni colonna. È necessario che per tutte le colonne non incluse nell'elenco sia previsto, dunque, il supporto di valori *NULL* o predefiniti. I valori specificati devono corrispondere alle colonne elencate. Il numero dei valori deve essere equivalente a quello delle colonne, mentre il tipo di dati, la precisione e la scala di ogni valore devono corrispondere a quelli previsti per i dati della colonna corrispondente. Perché l'operazione vada a buon fine è fondamentale che i dati presenti nel file siano in un formato compatibile a quello previste dalle tabelle. I formati definiti creando il database vengono scelti per ottimizzare l'esecuzione delle interrogazioni, dunque è necessario adattare i dati presenti nel file log a quelli previsti dalla struttura database: questa operazione è implementata pertanto nella fase di pre-elaborazione.

La progettazione si è sviluppata pensando a tutto ciò che si farà quando il database sarà stato creato. È sorto spontaneo, quindi, pensare al caricamento dei dati e sono state valutate, in particolare, due possibilità: caricare tutti i dati presenti in una linea del file log e poi estrarre in ambiente *SQL* tutte le informazioni relative ai singoli oggetti (quindi non effettuare alcun tipo di pre-elaborazione), oppure separare, prima del caricamento nel database, tutti gli oggetti e inserirli nella struttura di memorizzazione già correlati di tutte le informazioni ricavabili (cioè effettuare una pre-elaborazione completa). La scelta fatta è stata di effettuare in fase di pre-elaborazione una prima classificazione che consiste nel separare tutti i campi previsti dal *Common Log File Format* e caricarli separatamente.

2.1.1. Perl

Nella fase di pre-elaborazione si è scelto di usare il linguaggio di programmazione *Perl*[17] (acronimo di *Practical Extraction* and *Report Language*). È un linguaggio di scripting creato nel 1987 da Larry Wall[18]. *Perl* è stato creato inizialmente come ausilio ai sistemisti, con funzioni di manipolazione del testo e dei file. Si è evoluto tramite il sistema dei moduli in un linguaggio a carattere più generale, comprendente l'elaborazione di immagini, l'interrogazione di banche dati e i processi di comunicazione via rete. Ciò è reso possibile pure dal fatto che si può integrare codice scritto in *C* in un programma *Perl*. A partire dalla versione 5, il linguaggio stesso permette (ma non obbliga) di scrivere programmi orientati agli oggetti. *Perl* è comunemente ritenuto un linguaggio interpretato, ossia che per essere eseguito viene interpretato al momento dell'esecuzione. In realtà, la prima cosa che fa l'interprete è di trasformare il codice sorgente in un grafo intermedio. Questo approccio permette di limitare la lentezza tipica dei linguaggi interpretati. Nato in ambiente *Unix* e distribuito contemporaneamente con due licenze liberali, è disponibile anche per i sistemi operativi *Windows* e *MacOs* precedente alla versione *MacOSX* (che appartiene alla famiglia *Unix*). L'implementazione per *Windows* più diffusa viene distribuita da una società, la *ActiveState*, che da un lato offre con una licenza open source moduli specifici per il sistema operativo *Windows*, dall'altro vende degli ambienti di sviluppo integrati, sia per *Perl* che per altri linguaggi open source quali ad esempio *Python* e *Tcl*. Benché si ritenga che sia una delle grandi novità nel campo della programmazione, il giudizio su *Perl* da parte della comunità di programmatori è vario. Viene giudicato negativamente per il fatto che facilita la scrittura di programmi difficili da leggere rendendo complicata la loro manutenzione (al punto che il nome del linguaggio è stato re-interpretato come *Pathologically Eclectic Rubbish Lister*); ma viene apprezzato per la facilità di scrivere programmi potenti, ma semplici, e per la libertà semantica che lascia al programmatore, tanto che "non c'è un unico modo di fare le cose" è uno dei modi di dire legati a *Perl*. Wall - linguista - ritiene questa libertà semantica un pregio, proprio perché più simile al linguaggio umano. Un ulteriore aspetto positivo che attrae i programmatori è l'ampia disponibilità di moduli rilasciati con licenze open source, quasi sempre le stesse di *Perl*. Moduli solitamente documentati bene, in quanto il linguaggio stesso offre il *Pod*, un modo di includere la documentazione all'interno del codice, garantendo così che essa sia assieme al modulo stesso. La comunità ha creato una rete di repository, chiamata *CPAN*[19] (*Comprehensive Perl Archivi Network*), nella quale i moduli contribuiti dagli autori vengono classificati ed

[17] http://www.perl.com/, http://www.perl.org/
[18] L. Wall, T. Christiansen, e R.L. Schwartz, *Programming Perl*, O'Reilly Associates, seconda edizione, 1996.
[19] http://www.cpan.org/

organizzati per la distribuzione. I moduli stessi sono anche archiviati in *CPAN*, oltre che sui siti scelti dai loro autori. Infine, in quanto linguaggio interpretato e dunque sempre distribuito con il codice sorgente visibile, favorisce la pratica liberale. Una delle caratteristiche più importanti di *Perl* sono le espressioni regolari, che permettono la ricerca e la sostituzione di stringhe di testo descritte con caratteri speciali. Le espressioni regolari non sono un'esclusiva di *Perl*, e anzi sono state derivate da Wall a partire da espressioni già esistenti, ma quelle di *Perl* sono riconosciute come le più potenti tanto che numerosi linguaggi le emulano mediante funzioni di libreria o moduli esterni. Il linguaggio e l'interprete vengono sviluppati da un gruppo di sviluppatori volontari, guidati da Wall.

Date le consistenti dimensioni, occorre minimizzare i tempi di elaborazione di ogni dato, specie delle operazioni più frequenti quali le manipolazioni di stringhe. Pertanto *Perl* risulta particolarmente adatto nell'elaborazione dei file log. Si ricorda inoltre che tutti i programmi *Perl* creati dell'ambito di questo progetto sfruttano *Pod* per la documentazione.

2.1.2. Formato del file

La prima fase di elaborazione è stata conformare il file di log in un formato adatto alle operazioni successive. È stato sviluppata a tale scopo il programma *formatoLog*.
Questo programma prevede tre parametri in input.

Parametro	Descrizione	Default
File input.	Stringa che indica il nome del file di input che deve essere in un formato previsto da Apache.	`AccessLog`
File output	Stringa che indica il nome del file di output.	`FormatoLog`
Formato linee	Stringa rappresentante il formato delle linee presenti nel file di input.	`%h %l %u %t \"%r\" %>s %b`

Tabella 4 – Parametri *formatoLog*

L'estensione dei file indicati nei parametri non deve essere indicata in quanto devono essere di tipo *txt*.

I server web permettono di scrivere nei propri log di accesso una serie di informazioni relative ad ogni richiesta fatta via HTTP. Nel caso di *Apache*, ad esempio, con la direttiva di configurazione *LogFormat* si assegna un nickname ad un possibile tracciato record del log e si decide il formato dei dati che deve contenere.

I dati che possono essere scritti nel log vengono identificati con lettere o stringhe precedute dal simbolo %:

Identificatore	Descrizione
`%b`	Dimensioni in byte del file trasferito (coincide con l'header Content-Length).
`%f`	Nome del file e path completo del documento richiesto.
`%h`	Hostname o, se non viene risolto, indirizzo IP del client.
`%a`	Indirizzo IP del client. È uguale a *%h* se l'hostname non viene risolto.
`%A`	Indirizzo IP del server.
`%l`	Nome dell'utente remoto, se fornito da un idented lookup
`%p`	Porta TCP del server, a cui è arrivata la richiesta del client (di solito 80).
`%P`	PID del child di Apache che ha gestito la richiesta.
`%r`	Prima riga della richiesta HTTP,che contiene il metodo usato (GET, POST...).
`%s`	Status code HTTP della risposta.
`%t`	Data e ora della richiesta. Personalizzabile con *%{formato}t*.
`%T`	Numero di secondi impiegati da Apache per processare la richiesta.
`%u`	Nome dell'utente eventualmente autenticato sul server .
`%U`	URL richiesta dal client. Contenuta anche in *%r*.
`%v`	*Canonical Server Name*, come definito nella direttiva *ServerName*.
`%V`	*Server Name* secondo quanto definito in *UseCanonicalName*.

Tabella 5 – Identificatori formato dei log – campi semplici

Identificatore	Descrizione
`%{Variabile}e`	Variabile d'ambiente, così come definita dal server.
`%{Header}i`	Header HTTP nella richiesta del client (esempi comuni: *%{User-Agent}i* *%{Referrer}i*).
`%{Header}o`	Header HTTP nella risposta del server (es. : *%{Last-Modified}o*).
`%{Nota}n`	Stringa che può essere scambiata fra il core di Apache e un modulo (esempio tipico, per il *mod_usertrack*: *%{Cookie}n*).

Tabella 6 – Identificatori formato dei log – campi complessi

Se nella stringa sono presenti più campi dello stesso tipo, nell'elaborazione del file di input verrà preso in considerazione solo il contenuto della prima occorrenza di tale campo; nel caso non

venga indicato il terzo parametro, viene assunto come tale il valore di default, corrispondente al formato standard *Common Log File Format*.

Il programma si occupa di elaborare ogni linea del file input: per ogni riga letta da input viene creata una nuova linea da scrivere nel file output; questa nuova linea è costituita da tutti i tipi di campo possibili (elencati precedentemente) separati dal carattere *pipe* ('|') e concatenati nel seguente ordine:
%h|%l|%u|%t|%r|%s|%b|%i|%f|%a|%A|%p|%P|%T|%U|%v|%V|%e|%o|%n, ottenuto considerando dapprima i campi previsti dal *Common Log File Format*, seguiti dai restanti tipi di campo "semplici" ordinati alfabeticamente, concatenati ai tipi di campo "complessi", anch'essi ordinati alfabeticamente; nella linea che viene scritta in output il contenuto di ogni campo è quello della linea letta dal file input, nel caso essa contenga quel tipo di campo; nel caso in cui il tipo di campo non sia presente nel parametro indicante il formato delle linee di input, il contenuto ad esso relativo nella linea di output è quello di default ('-').

2.1.3. Formato dei dati

La fase successiva consiste nel rendere il formato dei dati presenti nel file di log idoneo all'inserimento nel database. Questo compito è risolto dal programma *elaboraLog* che prevede due parametri in input analoghi a quelli del programma precedente.

Parametro	Descrizione	Default
File input	Stringa che indica il nome del file di input che deve essere del tipo fornito dal programma *formatoLog*.	`FormatoLog`
File output	Stringa che indica il nome del file di output.	`ElaboraLog`

Tabella 7 – Parametri *elaboraLog*

L'estensione dei file indicati nei parametri non deve essere indicata in quanto devono essere di tipo *txt*.

Per ogni linea letta dal file input vengono eseguite le seguenti operazioni:

- campo richiesta:
  - vengono divisi i tre campi *metodo*, *url* e *protocollo* dal separatore *pipe* ('|');
  - se uno o più campi non sono presenti vengono inseriti con valore di default ('-');
- campo *data*:
  - i caratteri che indicano il mese vengono sostituiti con due caratteri numerici (es. la stringa *Feb* viene sostituita dalla stringa *02*);
  - vengono sostituiti i *due punti* (':'), presenti tra data e orario, con uno *spazio bianco* (' ');
  - così facendo il campo *data* risulta nel formato *datetime* utilizzato in SQL;
  - la data viene separata dal fuso orario con *pipe*;

La linea così modifica viene infine scritta nel file output.

2.1.4. Classificazione

L'ultima fase si occupa di separare tutti i dati memorizzandoli in file distinti e il programma *classiLog* si preoccupa di questo. Esso prevede dieci parametri in input.

Parametro	Descrizione	Default
File input	Stringa che indica il nome del file di input che deve essere del tipo fornito dal programma *elaboraLog*.	`ElaboraLog`
File Client	Stringa che indica il nome del file di output relativo all'entità *Client*.	`ClientLog`
File Account	Stringa che indica il nome del file di output relativo all'entità *Account*.	`AccountLog`
File Data	Stringa che indica il nome del file di output relativo all'entità *Data*.	`DataLog`
File Richiesta	Stringa che indica il nome del file di output relativo all'entità *Richiesta*.	`RichiestaLog`
File Metodo	Stringa che indica il nome del file di output relativo all'entità *Metodo*.	`MetodoLog`
File Protocollo	Stringa che indica il nome del file di output relativo all'entità *Protocollo*.	`ProtocolloLog`
File Url	Stringa che indica il nome del file di output relativo all'entità *Url*.	`UrlLog`
File Linea	Stringa che indica il nome del file di output relativo all'entità *Linea*.	`LineaLog`
File Errore	Stringa che indica il nome del file di output nel quale vengono registrate le linee errate.	`ErrLog`

Tabella 8 – Parametri *classiLog*

L'estensione dei file indicati nei parametri non deve essere indicata in quanto devono essere di tipo *txt*.

I campi di ogni linea letta sono (per ipotesi) divisi l'uno dall'altro dal carattere *pipe*; durante l'elaborazione di ogni linea vengono estratti i valori dei campi d'interesse; questi vengono ulteriormente elaborati in funzione della struttura di un database nel quale verranno inseriti. Infine i valori elaborati vengono distribuiti nei rispettivi file.

I campi elaborati sono i seguenti:

Client

Componenti	Default
Indirizzo IP	`100.10.10.10`
Hostname	`-.-.-`
Dominio	`-`

Tabella 9 – Campo *Client*

- viene verificato il tipo di informazione contenuta nella linea (IP o hostname);
- nel caso il dato sia un hostname viene estratto il valore del dominio;
- alle informazioni non rilevate da input vengono assegnati i valori di default;
- i tre campi vengono concatenati in modo ordinato e separati dal carattere *pipe*: *indirizzoIp*|*hostname*|*dominio*

Rfc

Componenti	Default
Rfc	`-`

Tabella 10 – Campo *Rfc*

Account

Componenti	Default
Account	`-`

Tabella 11 – Campo *Account*

Data

Componenti	Default
Data e orario	`00/00/00` `00:00:00`
Fuso orario	`+00000`

Tabella 12 – Campo *Data*

- i valori vengono separati dal carattere *pipe*: *data orario*|*fusoOrario*

Richiesta

Componenti	Default
Metodo	-
Url	-
Protocollo	-/-.-

Tabella 13 – Campo *Richiesta*

- i valori vengono separati dal carattere *pipe*: *metodo|url|protocollo*

Metodo

Componenti	Default
Metodo	-

Tabella 14 – Campo *Metodo*

Url

Componenti	Default
Url	-

Tabella 15 – Campo *Url*

Protocollo

Componenti	Default
Nome	-
Versione	-/-.-

Tabella 16 – Campo *Protocollo*

- se la versione non è rilevabile in input gli viene assegnato il valore di default;
- i valori vengono separati dal carattere *pipe*: *nome|versione*

Stato

Componenti	Default
Stato	000

Tabella 17 – Campo *Stato*

Dimensione

Componenti	Default
Dimensione	-

Tabella 18 – Campo *Dimensione*

Linea

Componenti	Default
Client	`100.10.10.10`
Rfc	`-`
Account	`-`
Data	`00/00/00` `00:00:00|+00000`
Richiesta	`-|-|-/-.-`
Stato	`000`
Dimensione	`-`

Tabella 19 – Campo *Linea*

- contiene tutti i valori dei campi a seguito dell'elaborazione, separati dal carattere *pipe*: *client*|*rfc*|*account*|*data*|*richiesta*|*stato*|*dimensione*

Ognuno dei campi viene inserito in un file diverso.

2.2. Definizione della struttura del database

2.2.1. Caratteristiche comuni

Prima della creazione delle classi sono stati definiti dei tipi di dato, in modo da avere nel database valori omogenei, tutti nei formati definiti o nel formato *datetime*. Di seguito un breve commento ai tipi di dato definiti:

- *id_log*: tipo di dato intero, non nullo, utilizzato per identificare univocamente oggetti;
- *nomeCorto_log*: stringa di pochi caratteri, al massimo dieci, utilizzato per rappresentare brevi codici numerici o alfanumerici;
- *nome_log*: stringa lunga al massimo cento caratteri, usata per rappresentare nomi;
- *nomeLungo_log*: stringa che arriva fino a duecento caratteri, utilizzata soprattutto per brevi descrizioni;
- *nomeMoltoLungo_log*: stringa particolarmente lunga, può contenere fino a quattrocento caratteri, nata per rappresentare descrizioni ma sfruttata anche per rappresentare oggetti (ad es. URL) che eccezionalmente si presentano in formato non standard.

Per ciascuna entità è stata scelta una lettera dell'alfabeto identificativa, presente in coda al nome di ogni attributo dell'entità stessa. Ciò è utile nel momento in cui si scrivono interrogazioni, infatti permette di denominare in modo analogo ma diverso, attributi che si riferiscono a classi differenti. In tal modo è possibile richiamare tali attributi senza specificare l'entità di appartenenza. Ad esempio si considerino l'entità *Client* e l'entità *Server*: in entrambe deve essere presente un attributo che rappresenti l'indirizzo IP. Si potrebbe avere l'entità *Client* con attributo *ip* e l'entità *Server* con lo stesso attributo *ip*. Nel caso si volessero selezionare entrambi gli *ip* in un'interrogazione occorrerebbe specificare l'entità dell'attributo: *select client.ip, server.ip, from.…*Utilizzando la notazione scelta, invece, la scrittura della query risulta semplificata: *select ipC, ipS, from.…*.

Ogni entità, escluse quelle popolate precedentemente al caricamento dei file di log (entità *Dominio*, entità *Formato*, entità *Errore* e entità *Stato*), ha un attributo che indica la correttezza dell'istanza. Questo riferimento all'entità *Errore* esprime la presenza di anomalie, imperfezioni o la correttezza dell'oggetto memorizzato. In ogni entità l'idea di errore assume un significato diverso e sarà spiegato, di volta in volta, ciò che esso può esprimere. I dati contenuti nelle classi in cui non è presente tale attributo, invece, sono considerati sempre corretti.

2.2.2. Classi principali

2.2.2.1. Entità Linea

La prima scelta effettuata riguarda quali linee del file log considerare e quali dati di ogni linea memorizzare nel database. Si è deciso di memorizzare tutti i dati relativi a ciascuna linea in modo da rendere l'analisi il più possibile flessibile dal punto di vista utente: se anche attualmente non fossero considerati interessanti tutti i dati contenuti nel file, in futuro potrebbero diventarlo, è bene quindi registrare tutto delle linee considerate idonee all'analisi. Si rimanda, invece, al capitolo riguardante la fase di pre-elaborazione la definizione delle linee da includere nel caricamento. È nata spontaneamente, quindi, la definizione dell'entità *Linea*.

Per il momento si è scelto di considerare solo il formato *Common Log File Format* perciò è stato definito nell'entità un oggetto per ogni campo previsto da tale formato. In seguito, però, si potranno caricare dati relativi a linee in altri formati semplicemente aggiungendo attributi a questa entità.

Quasi tutti i campi sono riferimenti a oggetti di altre classi: *clientL* si riferisce all'entità *Client*, *accountL* all'entità *Account*, *dataL* e *fusoL* all'entità *Data*, *metodoL*, *richiestaL*, *protocolloL* e *versioneL* all'entità *Richiesta*, *statoL* all'entità *Stato*, *fileL* all'entità *NomeFile*, *serverL* all'entità *Server* e *errL* all'entità *Errore*. Dunque per chiarimenti sul significato di tali valori vedere i paragrafi successivi.

idL è utilizzato per non perdere traccia dell'ordine in cui sono state inserite le linee nel log mentre *fileL* e *serverL* servono per rintracciare il file a cui apparteneva la linea.
errL può indicare che mancano alcuni campi, o che alcuni campi non sono del tutto corretti.

Attributi dell'entità:

Nome	Formato
idL	id_log identity
clientL	nome_log
rfcL	nomeCorto_log
accountL	nome_log
dataL	datetime
fusoL	nomeCorto_log
metodoL	nomeCorto_log
richiestaL	nomeMoltoLungo_log
protocolloL	nomeCorto_log
versioneL	nomeCorto_log
statoL	nomeCorto_log
dimL	nomeCorto_log
fileL	nomeLungo_log
serverL	nome_log
errL	id_log

Tabella 20 – Attributi entità *Linea*

Esempio:

idL = *14*

clientL = 154.86.47.75

rfcL = -

accountL = *danny78*

dataL = 25/09/2005 14:39:23

fusoL = *-0600*

metodoL = *POST*

richiestaL = /dbgroup/index.html

protocolloL = *HTTP*

versioneL = *1.1*

statoL = *304*

dimL = *2938*

fileL = accessLog_25_09_05

serverL = 173.456.45.32

errL = *11*

2.2.2.2. Entità Richiesta

La linea di richiesta HTTP è composta da metodo, URI e versione del protocollo: l'entità *Richiesta* contiene dunque un riferimento a un'istanza dell'entità *Url* (*nomeR*) e un riferimento all'entità *Metodo* (*metodoR*, *protocolloR* e *versioneR*).

errR può essere utilizzato per indicare che non è prevista o che è anomala l'applicazione del metodo indicato alla risorsa cui si fa riferimento.

Attributi dell'entità:

Nome	Formato
nomeR	nomeMoltoLungo_log
metodoR	nome _log
protocolloR	nomeLungo_log
versioneR	nomeCorto_log
errR	id_log

Tabella 21 – Attributi entità *Richiesta*

Esempio:

nomeR = http://www.unimore.it/servizi/residenze.asp

metodoR = *GET*

protocolloR = *HTTP*

versioneR = *1.0*

errR = *3*

2.2.2.3. Entità Client

Un host è un qualsiasi terminale collegato ad Internet. Gli host possono essere di diverso tipo, ad esempio computer, palmari, dispositivi mobili e così via, fino a includere web TV, dispositivi domestici e thin client. L'host è definito in questo modo perché *ospita* programmi di livello applicativo che sono sia client (ad esempio browser, reader di posta elettronica), sia server (ad esempio, web server). Uno stesso host può agire contemporaneamente da client e da server, in particolare con le applicazioni peer to peer (esempio *Napster*, *Kazaa*, etc.).

L'entità C*lient* e l'entità *Server*, dunque, sono del tutto analoghe e raccolgono informazioni relativamente ad un host, che si comporta rispettivamente da client o da server. i*pC/S* rappresenta l'indirizzo IP. Nel file log non è sempre presente un indirizzo IP del client: nel caso sia presente un hostname esso viene risolto nella fase di pre-elaborazione. Il formato dell'attributo rispecchia il formato di un indirizzo IP v. 4 ad esempio: *192.80.54.74*. In futuro si potrebbero fare altre scelte a proposito di questo campo:

- si potrebbe rafforzare il vincolo del formato e considerare solo le classi IP effettivamente utilizzate in rete
- si potrebbe ricavare da questo campo, il valore di un altro oggetto che rappresenti la classe di indirizzi IP a cui appartiene quello considerato
- si potrebbero considerare gli indirizzi IP v. 6.
- Per maggior chiarezza si veda il paragrafo successivo *Indirizzi IP*.

L'eventuale hostname (*nomeC/S*) risolto dal DNS è invece costituito da almeno tre stringhe separate da *punti* e *dataC/S* indica la validità dell'informazione fornita dal DNS. L'attributo *dominioC/S* si riferisce all'hostname.

errC può indicare un indirizzo IP non valido (es. *999.23.888.33*), inesistente, oppure che la data di validità del dell'hostname è scaduta (*dataC/S* passata).

Attributi dell'entità *Client*:

Nome	Formato
ipC	nome_log check(ipC like '[1-9]%[.][0-9]%[.][0-9]%[.][0-9]%'
nomeC	nome_log check(nomeC like '%[.]%[.]%')
dominioC	nomeCorto_log
dataC	datetime
errC	id_log

Tabella 22 – Attributi entità *Client*

Esempio:

ipC = 192.68.72.72

nomeC = www.ing.unimo.it

dominioC = *it*

dataC = 31/09/2005

errC = *2*

2.2.2.4. Indirizzi IP

L'indirizzamento IP permette di identificare ogni host all'interno di una rete TCP/IP. Grazie all'utilizzo delle classi di indirizzi ed al subnetting è possibile organizzare e gestire in modo più efficiente il proprio network.

Un indirizzo IP, chiamato anche indirizzo logico, rappresenta un identificativo software per le interfacce di rete e viene utilizzato in combinazione con l'indirizzo fisico (MAC), il quale consente di determinare in modo univoco ogni interfaccia di un dispositivo di rete. Un IP Address è un numero di 32 bit suddiviso in quattro gruppi da 8 bit ciascuno: la forma con la quale viene solitamente rappresentato è detta decimale puntata *(Dotted Decimal*).

Essendo ogni numero rappresentato da 8 bit, può assumere un range di valori da 0 a 255. In realtà esistono alcuni indirizzi particolari, di conseguenza non tutti i valori sono disponibili al fine di identificare un host nella rete.

Esistono alcuni particolari indirizzi di rete che non possono essere assegnati per l'identificazione di un host, tra questi abbiamo: *network*, *broadcast* e *loopback*:

- *Network*: quando il valore che rappresenta l'host ha valore 0, l'indirizzo è detto di rete o Network Address, es. : *192.168.5.0*;
- 0.0.0.0: quando tutti i bit hanno valore zero, identificano "questo host";
- *Broadcast*: quando il numero che rappresenta l'host ha valore 255, l'indirizzo è detto di broadcast o broadcast address, e rappresenta tutti gli host di quella rete. Inviare un pacchetto all'indirizzo *192.168.5.255* equivale a mandare un pacchetto a tutti gli host della rete *192.168.5*;
- *Broadcast di rete*: abbiamo questo tipo di indirizzo quando tutti i numeri, sia della parte relativa all'host sia della parte relativa alla rete hanno valore 255. Inviare un pacchetto a *255.255.255.255* significa inoltrarlo verso tutti gli host della rete corrente;
- *Loopback*: è utilizzato per funzioni di test del protocollo TCP/IP, non genera traffico di rete e corrisponde all'indirizzo 127.0.0.1.

Per consentire una migliore organizzazione della rete, gli indirizzi disponibili sono stati suddivisi in classi in base alle dimensioni del network da gestire. In questo modo verranno utilizzate le classi più adatte alle dimensioni delle reti, con conseguente minore spreco di indirizzi IP. Sono disponibili cinque classi di indirizzi IP, di cui solo le prime tre possono essere utilizzate per assegnare indirizzi agli host.

- *Classe A:* è rappresentata da indirizzi di tipo *Rete.Host.Host.Host* ovvero 8 bit per identificare la rete (di cui il primo fisso) e 24 per identificare gli host;
- *Classe B*: è rappresentata da indirizzi di tipo: *Rete.Rete.Host.Host* ovvero 16 bit per identificare la rete(di cui i primi due fissi) e 16 per identificare gli host;
- *Classe C*: è rappresentata da indirizzi di tipo: *Rete.Rete.Rete.Host* ovvero 24 bit per identificare la rete (di cui i primi tre fissi) e 8 per identificare gli host;
- *Classe D:* sono indirizzi di rete riservati ai gruppi multicast e non assegnabili ai singoli host;
- *Classe E*: sono indirizzi riservati per usi futuri.

Sono stati definite alcune classi di indirizzi, definiti nella RFC 1918, chiamati privati, per le reti locali che non accedono ad internet:

- da 10.0.0.0 a 10.255.255.255;
- da 172.16.0.0 a 172.31.255.255;
- da 192.168.0.0 a 192.168.255.255;

Questi indirizzi non possono essere utilizzati in Internet, e sono riservati per utilizzi in reti interne. Qualora però un host all'interno di un LAN (rete locale) si connetta ad internet il suo indirizzo verrà riscritto tramite NAT (*Network Address Translation*) da un router od una macchina che fa da gateway verso Internet.

2.2.3. Le altre classi

2.2.3.1. Entità Errore

L'entità *Errore* è stata creata per indicare la correttezza dei dati. Ogni errore è correlato da una breve descrizione. Gli errori, inoltre, sono raggruppati in livelli di priorità che ne indicano la gravità. La priorità è direttamente proporzionale alla gravità dell'errore, e priorità 0 significa assenza di errore. Il valore del campo ID, invece, non fornisce alcuna informazione relativa alla gravità dell'errore. Esso, infatti, viene assegnato, in modo incrementale, nel momento in cui viene inserita una nuova istanza, cioè quando è necessario distinguere un nuovo tipo di scostamento del dato rispetto ai valori previsti.
Attributi dell'entità:

Nome	Formato
idE	id_log
priorityE	id_log
descrizioneE	nomeLungo_log

Tabella 23 – Attributi entità *Errore*

Esempio:

idE = *100*

priorityE = *0*

descrizioneE = *Nessun errore*

2.2.3.2. Entità Dominio

Questa entità contiene brevi descrizioni dei TLD, cioè *Top-Level Domain*, più utilizzati. Questo termine inglese indica un dominio internet DNS di primo livello. I domini di primo livello[20], stabiliti dall'ICANN, sono identificati dai suffissi degli indirizzi alfanumerici; per esempio, l'indirizzo internet dell'Università degli Studi di Modena e Reggio Emilia è *www.unimore.it* e quindi ricade all'interno del dominio di primo livello *.it***.** Al momento della loro creazione, nel 1985, i TLD ufficialmente riconosciuti erano *.arpa* (il primo dominio creato, in seguito ritirato e utilizzato solamente per pseudodomini)*, .com, .edu, .gov, .net, .mil, .org*. Furono poi aggiunti *.int, .aero, .biz, .coop, .info, .name, .pro*. Si definiscono ccTLD (*Country Code Top Level Domains*) i domini di primo livello corrispondenti ai suffissi degli stati sovrani riconosciuti tali,

[20] http://bertola.eu.org/icfaq/domini.htm

che consistono nella sigla internazionale ISO 3166-2 del paese (*.it* per l'Italia, *.de* per la Germania ecc.) e l'ormai approvato *.eu* per l'Unione Europea, che dovrebbe essere utilizzato a partire dal 2006.
Attributi dell'entità:

Nome	Formato
nomeO	nomeCorto_log
descrizioneO	nome_log

Tabella 24 – Attributi entità *Dominio*

Esempio:

nomeO = *it*

descrizioneO = Italia

2.2.3.3. Entità Stato

La linea di stato creata dal protocollo HTTP riporta un codice a tre cifre catalogato nel seguente modo:

- *1xx* : Informazioni
- *2xx*: Successo
- *3xx*: Invio ad altro URI
- *4xx*: Errore client
- *5xx*: Errore server
- *6xx*: Errore non standard

Nel caso più comune il server risponde con un codice *200* (Ok), altri casi comuni sono:

- *302* (Trovato): la risorsa è raggiungibile con un altro URI. Di norma i browser eseguono la richiesta all'URI indicato in modo automatico senza interazione dell'utente.
- *404* (Non trovato): la risorsa richiesta non è stata trovata e non se ne conosce l'ubicazione. Di solito avviene quando l'URI indicato in modo incorretto od è stato rimosso il contenuto dal server.
- *500* (Errore server interno): il server non è in grado di rispondere alla richiesta a causa di un suo problema interno.
- *600* (Errore non standard): l'errore dipende dalla definizione che è stata scelta nell'ambito dell'applicativo che lo ha fornito.

L'entità S*tato* contiene la descrizione di ognuno di questi codici, per cui ogni istanza è costituita da una breve stringa rappresentante il codice (*nomeT*) e da una stringa più lunga per la descrizione (*descrizioneT*).
Attributi dell'entità:

Nome	Formato
nomeT	nomeCorto_log
descrizioneT	nomeLungo_log

Tabella 25 – Attributi entità *Stato*

Esempio:

nomeT = *304*

descrizioneT = *Non modificato*

2.2.3.4. Entità Formato

In informatica, un formato di file è la convenzione che viene usata per leggere, scrivere e interpretare i contenuti di un file. Poiché i file non sono altro che insiemi ordinati di byte, cioè semplici numeri, per poter associare al loro contenuto cose diverse, si usano convenzioni che legano i byte ad un significato. Il formato di un certo file è comunemente indicato attraverso l'estensione, che è una serie di lettere (in genere tre, per motivi storici) unita al nome del file attraverso un *punto*. Ad esempio, "prova.txt" è un file di testo, mentre "prova.jpg" è un'immagine. Per molti formati sono state pubblicate delle specifiche[21] che descrivono esattamente come i dati devono essere codificati. Non sempre tali specifiche sono disponibili: innanzitutto alcuni formati sono considerati segreti industriali e le loro specifiche non vengono distribuite pubblicamente; inoltre in molti casi gli sviluppatori non scrivono un documento di specifica separato, ma definiscono il formato solo implicitamente attraverso il programma che lo gestisce. Esistono molti formati diversi, e altri ne vengono creati di tanto in tanto.

L'entità *Formato*, contiene la descrizioni di quelli più diffusi.
Attributi dell'entità:

Nome	Formato
nomeF	nomeCorto_log
descrizioneF	nomeLungo_log

Tabella 26 – Attributi entità *Formato*

[21] http://www.ace.net.nz/tech/TechFileFormat.html

Esempio:

nomeF = *txt*

descrizioneF = *File di testo*

2.2.3.5. Entità Protocollo

Un protocollo è un insieme di regole che definiscono il formato dei messaggi scambiati e consentono a due o più macchine o host di comunicare tra di loro e di comprendere la comunicazione. Nei file log vengono tracciate informazioni riguardante messaggi scambiate tramite il protocollo HTTP. HTTP è l'acronimo di *HyperText Transfer Protocol* (protocollo di trasferimento di un ipertesto) ed è usato come principale sistema per la trasmissione di informazioni sul web. Le specifiche del protocollo sono attualmente in carica al W3C (*World Wide Web Consortium*). La prima versione effettivamente disponibile del protocollo fu la HTTP/1.0. In seguito divennero evidenti alcuni limiti della versione 1.0 del protocollo e venne quindi esteso nella versione HTTP/1.1.

L'entità *Protocollo* rappresenta il protocollo utilizzato nella richiesta del log file con la relativa versione. Sia il nome del protocollo (*nomeP*) che la versione (*versioneP*) sono brevi stringhe. Teoricamente dovrebbe contenere al più due istanze, in quanto per il momento si registrano nei log solo le richieste HTTP e di questo protocollo si utilizzano solo due versioni. In futuro, però, potrebbero nascere nuove versioni del protocollo HTTP, o si potrebbero analizzare richieste di altri protocolli e, in tal caso, si potrà continuare a utilizzare questa entità per memorizzare il protocollo usato nella richiesta.

errP, l'intero che identifica l'errore, potrà essere utilizzato per indicare che una versione di un protocollo non viene utilizzata, oppure non è riconosciuta dal sistema di analisi.
Attributi dell'entità:

Nome	Formato
nomeP	nomeCorto_log
versioneP	nomeCorto_log
errP	id_log

Tabella 27 – Attributi entità *Protocollo*

Esempio:

nomeP = *HTTP*

versioneP = *1.1*

errP = *0*

2.2.3.6. Entità Metodo

Il protocollo HTTP si basa su una serie di comandi diretti al server per richiedere o inviare dati. Tra questi, i tre comandi più noti sono:

- *GET*: tramite questo comando viene richiesta una risorsa sul server. Una risorsa può essere rappresentata da una pagina HTML, un'immagine o un qualsiasi altro tipo di file, ma può essere costituito anche dal risultato dell'esecuzione di uno script o di una pagina ASP, PHP, ecc.
- *POST*: il comando POST consente di inviare dati al server contestualmente alla richiesta di una risorsa. Generalmente questo comando viene utilizzato per inviare dati raccolti tramite una form.
- *HEAD*: con questo comando vengono richieste al server informazioni su una risorsa, non la risorsa stessa. Ad esempio, può essere richiesta la data dell'ultima modifica, le dimensioni o il tipo.

Le specifiche del protocollo prevedono altri comandi, come *PUT*, *DELETE*, *CONNECT*, ecc., e l'entità *Metodo* rappresenta proprio questi comandi.

errM può indicare che il metodo non è conosciuto dal sistema di analisi o che non ne è prevista l'implementazione dal protocollo a cui si riferisce.

Attributi dell'entità:

Nome	Formato
nomeM	nomeCorto_log
protocolloM	nomeCorto_log
versioneM	nomeCorto_log
errM	id_log

Tabella 28 – Attributi entità *Metodo*

Esempio:

nomeM = *GET*

protocolloM = *HTTP*

versioneM = *1.1*

errM = *0*

2.2.3.7. Entità Url

Ogni istanza *Url* rappresenta un *Uniform Resource Locator*, cioè l'indirizzo di una risorsa in Internet. Ogni *URL* è composto dalle seguenti parti principali:

- lo *schema* o protocollo, utilizzato per indirizzare la risorsa
- il *nome* dell'host o server, oppure un nome di dominio
- il *path* o nome file della risorsa

Esempio: http://www.ing.unimo.it/index.html

- *http*: schema o protocollo (HTTP);
- *www.ing.unimo.it*: nome del server (*it*), completo di dominio (*unimo*);
- *index.html*: pathname completo della risorsa (una pagina web).

In Pagine Attive (ASP o PHP) gli URL possono anche contenere una *query-string*, preceduta dal carattere *punto interrogativo* ('?') e che serve per trasmettere ulteriori informazioni al server sul percorso da seguire attingendo a risorse del database (es. *: http://www.nomesito.it/paginaindice.asp?sezione=idSezione).* Nelle richieste registrate dal file log si fa sempre riferimento ad una risorsa, indicando un URL completo, o indicando solo una parte dell'URL nel caso in cui il pathname completo (es. : *http://www.ing.unimo.it/index.html*) venga sostituito da un path relativo (es. : */index.html*), nel caso in cui, cioè, non venga indicato l'host o il server.

Le istanze *Url* ripropongono dunque l'indirizzo per accedere ad una risorsa in rete e il formato di tale risorsa. La stringa *nomeU* è una stringa piuttosto lunga in quanto talvolta nei file di log sono presenti URL seguiti da vari parametri e/o comandi, il che rende l'indirizzo esteso. L'attributo *dominoU* si riferisce al dominio dell'host o del server (se presente).

errU può indicare la presenza di parametri nell'istanza, un formato dell'URL non riconosciuto o parametri non identificati dal sistema di analisi.

Attributi dell'entità:

Nome	Formato
nomeU	nomeMoltoLungo_log
tipoU	nomeCorto_log
dominioU	nomeCorto_log
errU	id_log

Tabella 29 – Attributi entità *Url*

Esempio:

nomeU = http://www.unimore.it/servizi/residenze.asp

tipoU = *asp*

dominioU = *it*

errU = *1*

2.2.3.8. Entità Data

L'entità *Data* rappresenta l'istante in cui viene effettuata la richiesta con precisione di un secondo. L'attributo *dataD* rappresenta l'oggetto nel formato *gg/mm/aaaa hh:mm:ss*, dove *gg* indica il giorno del mese, *mm* indica il mese, *aaaa* indica l'anno, *hh* indica l'ora, *mm* indica i minuti e *ss* indica i secondi. Questi valori sono rappresentati anche singolarmente nei relativi attributi. *fusoD* rappresenta, invece, la differenza dell'orario rispetto a quello di Greenwich, mentre *giornataD* indica il giorno della settimana (lunedì, martedì,…).

errD può indicare una data futura.

Attributi dell'entità:

Nome	Formato
dataD	datetime
fusoD	nomeCorto_log
annoD	datename(year,dataD)
meseD	datename(month,dataD)
giornoD	datename(day,dataD)
giornataD	datename(weekday,dataD)
oraD	datename(hour,dataD)
munitiD	datename(minute,dataD)
secondiD	datename(second,dataD)
errD	id_log

Tabella 30 – Attributi entità *Data*

Esempio:

dataD = 21/09/2005 16:06:39

fusoD = *+0100*

annoD = *2005*

meseD = *Settembre*

giornoD = *21*

giornataD = *Mercoledì*

oraD = *16*

munitiD = *6*

secondiD =*39*

errD = *16*

2.2.3.9. Entità Account

L'entità *Account* serve per registrare le eventuali autenticazioni da parte di un utente. In alcuni casi, infatti, l'utente fornisce, di solito su richiesta, uno username per identificarsi. Questo oggetto è rappresentato da una stringa.

errA può indicare uno username non nel formato previsto (se sono stati definiti determinati formati, es. almeno otto caratteri), non riconosciuto o con account scaduto (o sospeso) nel caso si abbia una lista degli utenti aventi un account.
Attributi dell'entità:

Nome	Formato
nomeA	nome_log
errA	id_log

Tabella 31 – Attributi entità *Account*

Esempio:

nomeA = *mary76*

errA = *9*

2.2.3.10. Entità Server

L'entità *Server*, del tutto analoga all'entità *Client*, prevede tre attributi ulteriori che indicano la data in cui il server ha iniziato a essere coinvolto nell'attività di log, l'eventuale data in cui il monitoraggio è terminato e a quale sito appartengono le pagine presenti sulla macchina.

Il campo *errS* può quindi indicare un errore dovuto al fatto che *inizioS* sia una data futura, oppure che sia successiva alla data *fineS*, o che la data di validità DNS (*dataS*) è scaduta, che *ipS* non è un indirizzo IP esistente o che il sito (*sitoS*) non è riconosciuto.

Attributi dell'entità *Server*:

Nome	Formato
ipS	nome_log check(ipS like '[1-9]%[.][0-9]%[.][0-9]%[.][0-9]%'
nomeS	nome_log check(nomeS like '%[.]%[.]%')
dominioS	nomeCorto_log
dataS	datetime
inizioS	datetime
fineS	datetime
sitoS	nome_log
errS	id_log

Tabella 32 – Attributi entità *Server*

Esempio:

ipS = 100.50.29.29

nomeS = www.nomeserver.com

dominioS = *com*

dataS = 25/08/2005

inizioS = *10/01/2005*

fineS = 28/08/2005

sitoS = ServerOnLine

errS = *10*

2.2.3.11. Entità Evento

Questa entità è stata creata in previsione di utilizzi futuri. Grazie ad essa sarà possibile registrare eventi che potrebbero avere un impatto sul traffico in rete. Potrebbe capitare ad esempio, che il sito di una scuola abbia un picco di visitatori nella data in cui vengono pubblicati gli esiti di un esame, dunque risulterebbe utile ricondurre l'incremento anomalo di richieste all'evento.
Attributi dell'entità *Evento*:

Nome	Formato
idV	id_log
descrizioneV	nome_log
dataV	datetime

Tabella 33 – Attributi entità *Evento*

Esempio:

idV = *123*

descrizioneV = inizio vendita biglietti online

dataV = 22/09/2005

2.2.3.12. Entità NomeFile

Questa entità serve per istanziare i nomi dei file (*nomeN*) di log caricati nel database. *serverN* indica il server al quale si riferisce il file di log, in quanto ogni file registra richieste che arrivano ad un determinato server.

L'attributo *errN* può indicare un formato delle linee del file diverso da quello previsto (ad es. se in ogni linea non sono presenti tutti i campi del *Common Log File Format*, oppure se sono presenti campi ulteriori non previsti), o che il file non ha registrato tutte le richieste che avrebbe dovuto.

Attributi dell'entità *NomeFile*:

Nome	Formato
nomeN	nomeLungo_log
serverN	nome_log
errN	id_log

Tabella 34 – Attributi entità *NomeFile*

Esempio:

nomeN = accessLog_11_09_05

serverN = 180.48.58.58

errN = *8*

2.2.3.13. Chiavi primarie

Di ogni entità è stata definita una chiave primaria che specifica da quali attributi è resa unica ogni istanza. Le entità *Linea*, *Errore*, ed *Evento* sono identificate univocamente rispettivamente dagli attributi *idL*, *idE*, *idV*; il valore di questi attributi viene definito al momento dell'inserimento di ogni istanza: il sistema genera automaticamente numeri in sequenza crescente per le nuove righe inserite in una tabella creando valori univoci.

Alcune entità, invece, sono costituite da al più due campi (oltre al riferimento all'errore) uno dei quali identifica univocamente le istanze, mentre l'altro (eventuale) descrive l'istanza. Dunque due istanze possono avere la stessa descrizione, ma il nome che le distingue deve essere univoco. È il caso delle entità *Dominio*, *Stato*, *Formato*, e *Account* in cui sono definite chiavi primarie rispettivamente *nomeO*, *nomeT*, *nomeF*, *nomeA*.

La richiesta HTTP è costituita, oltre dall'URL, dal metodo e dal protocollo. In alcuni casi è utile analizzare solo gli indirizzi delle risorse (ad. esempio se si vuole sapere quali file sono stati coinvolti nelle richieste più frequentemente), quindi sarebbe inutile coinvolgere anche altre informazioni nell'analisi. Lo stesso vale per le altre due informazione della richiesta: da cui la creazione dell'entità *Metodo, Url* e *Protocollo*.

Nonostante si tenga traccia separatamente di questi tre elementi nelle relative entità è necessario tenere conto delle loro occorrenze insieme, in quanto la combinazione di un URL con un determinato metodo piuttosto che con un altro ha significato diverso (soprattutto nell'ambito dell'analisi). Utilizzare un URL col metodo GET, ad esempio, significa che si desidera ricevere il contenuto di quella risorsa, mentre utilizzare lo stesso URL col metodo POST significa inviare informazioni a quella risorsa. Dunque l'entità *Richiesta* è identificata da un riferimento all'entità *Url* e un riferimento all'entità *Metodo*.

Ciascun istanza di *Metodo* è identificata non solo dal nome (*nomeM*), ma anche da un riferimento al protocollo che lo implementa (*protocolloM* e *versioneM*). Molti comandi, quindi, potrebbero essere presenti più volte nell'entità: una volta per ogni versione del protocollo che li supporta.

Per quanto riguarda il protocollo si è notato i metodi di ciascuno non variano solo in base al nome di quest'ultimo, ma anche in relazione alla versione. Per questo motivo è preferibile tener separate queste due informazioni in modo tale da verificare l'integrità della relazione tra il metodo e il protocollo. È importante distinguere le versioni in quanto non supportano gli stessi metodi (solitamente la versione più recente supporta un numero maggiore di metodi) e talvolta implementano gli stessi metodi in maniera diversa. Per questo motivo la chiave primaria è definita dalla combinazione *nomeP* e *versioneP*.

La chiave primaria dell'entità *Url* è semplicemente *nomeU* in quanto ogni URL identifica univocamente la risorsa cui si riferisce. Analoga considerazione vale per le entità *Client* e *Server* in quanto ogni loro istanza rappresenta un dispositivo e ogni dispositivo è identificato univocamente dall'indirizzo IP: dunque le chiavi primarie risultano rispettivamente *ipC* e *ipS*.

La *Data* è caratterizzata sia dall'orario di un determinato giorno cui si riferisce, sia dalla differenza di fuso orario rispetto a Greenwich; tali dati vengono tenuti separati poiché la conoscenza dell'uno o dell'altro viene utilizzata in ambiti diversi (ad es. la data può essere utilizzata per individuare gli orari in cui si riceve un picco di richieste, il fuso, invece, per determinare in quale fasce giornaliere gli utenti accedono al sito nel caso si conosca la distribuzione geografica degli utenti.

Considerando l'entità *NomeFile* si osserva che ogni istanza è identificata dal nome del file (*nomeN*) e dal server (*serverN*) al quale arrivano le richieste registrate nel file. Ogni file, infatti, registra richieste relative ad un solo server. File diversi relativi allo stesso server devono avere necessariamente nomi differenti, mentre file che si riferiscono a server distinti possono avere lo stesso nome.

2.2.4. Schema relazionale

Dopo aver definito le scelte discusse nei paragrafi precedenti è stato possibile disegnare lo schema relazionale che rappresenta le entità e le relazioni fra esse considerate importanti nell'ambito di questo progetto.

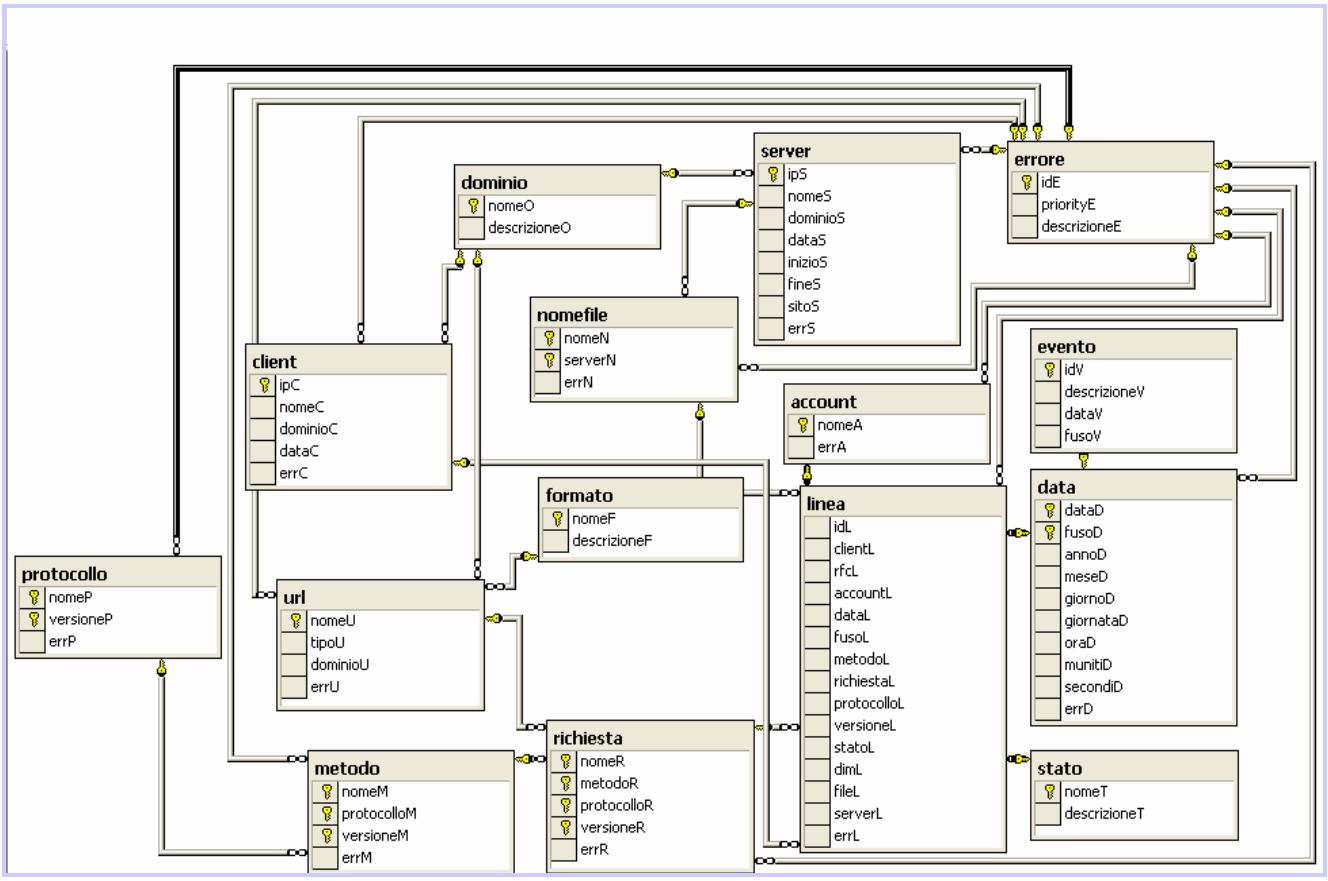


Figura 1 – Schema relazionale

2.3. Post-elaborazione

In seguito alla definizione della struttura database avviene la fase di caricamento. Come specificato in precedenza viene utilizzata l'istruzione *BULK INSERT*. I dati di ogni file (relativo ad ogni singola entità del database) vengono dapprima inseriti in una tabella temporanea. Successivamente vengono copiati dalla tabella temporanea a quella definitiva solo i valori mai istanziati e i valori i cui campi chiave non siano nulli. L'unica eccezione è prevista nella popolazione dell'entità *Linea*: in questo caso le istanze sono identificate da un ID creato all'atto dell'inserimento, quindi vengono inseriti nell'entità, eventualmente più volte, tutti i valori presenti nella tabella temporanea.

Dopo l'inserimento i dati possono essere ulteriormente elaborati. È in questa fase, ad e esempio, che può essere definita la *correttezza* di ogni istanza, cioè si può verificare, e identificare, l'eventuale presenza di errori nei dati raccolti.

2.3.1. Cursori

È stata implementata una procedura che identifica l'estensione degli URL (se ne hanno una). Si è scelto di effettuare queste operazioni in seguito all'inserimento dei dati così da riconoscere le solo estensioni presenti nella tabella relativa (tabella *Formato*), ma potrebbe essere effettuata anche nelle fase di pre-elaborazione.

Nella procedura viene utilizzato un *cursore*. A differenza dell'istruzione *select*, che restituisce un insieme di righe (tutte quelle che soddisfano le condizioni e le clausole specificate), i cursori permettono di elaborare una singola riga alla volta. Nella fase di post-elaborazione sono particolarmente utili in quanto si ha la necessità di analizzare (ed eventualmente modificare) un'istanza alla volta.

Nel caso della determinazione dei formati vengono considerate, una per una, tutte le righe dell'entità *Url*. Grazie al cursore si seleziona ogni singola riga: di ognuna si verifica se i caratteri presenti dopo l'ultimo *punto* (e prima di eventuali parametri) rappresentano un'istanza dell'entità *Formato*. In tal caso il formato dell'URL viene identificato, in caso non ci sia alcun match con la tabella *Formato* o non sia presente nessun *punto* nella riga selezionata il formato rimane indefinito ('-'). Analogamente si riconoscono i domini di URL e hostname, ma in tal caso il confronto sarà fatto con la tabella *Domini*.

3. Progetto e implementazione dei report più comuni

Dopo aver progettato e creato la struttura adatta per l'analisi dei file di log, e dopo avervi caricato i dati, si passa alla fase in cui vengono analizzati i contenuti di tali dati. Si iniziano dunque a produrre i primi risultati pratici. I dati che vengono forniti più frequentemente esprimono informazioni generali riguardo tutti i dati raccolti dal file log, tra cui:
il numero di richieste totali

- Quante richieste hanno avuto successo;
- Quante richieste sono fallite;
- Quante richieste sono state inoltrate ad altro URI o server;
- Quante linee del file sono corrotte;
- Quanti file sono stati richiesti;
- Quanti host hanno effettuato richieste;

e informazioni più specifiche relative alle sole richieste che hanno avuto successo, ad esempio:

- Quanti byte sono stati trasferiti;
- Quali pagine sono state più richieste;
- Quali browser sono stati utilizzati maggiormente per effettuare richieste;
- Quali sistemi operativi sono stati usati di più;
- Quali tipi di file sono stati richiesti;

Molti report rappresentano queste informazioni raggruppandole per intervalli temporali o temi d'interesse, ad esempio:

- Richieste effettuate ogni giorno o ogni ora;
- Richieste effettuate per ogni tipo di file;
- Richieste effettuate per ogni browser o sistema operativo riconosciuto dal sistema di analisi.

3.1. Definizione dei concetti presenti nei report

Per capire i dati mostrati dai report è fondamentale definire i vari termini utilizzati.

Per *richieste totali* si considerano tutte le richieste giunte al web server registrare nei file di log considerati, quindi questo termine indica il numero di linee presenti nei file. Per quanto riguarda il nostro sistema nella fase di pre-elaborazione vengono scartate alcune linee: quelle il cui formato non coincide con quello indicato per il file elaborato. Tramite le interrogazioni al database non è dunque possibile risalire a tali informazioni. Questa scelta è stata effettuata in quanto se i contenuti di una linea non corrispondono al formato indicato non è possibile attribuirvi alcun significato, non è possibile interpretare tali informazioni in alcun modo, quindi è inutile coinvolgerle nell'analisi.

Per *successo* si intende il caso in cui il codice di stato HTTP è del tipo *2xx* (caso in cui viene inoltrato il documento richiesto) oppure vale *304* (caso in cui il documento è stato richiesto pur se non necessario in quanto non è stato modificato dall'ultima volta in cui il client lo ha ricevuto).

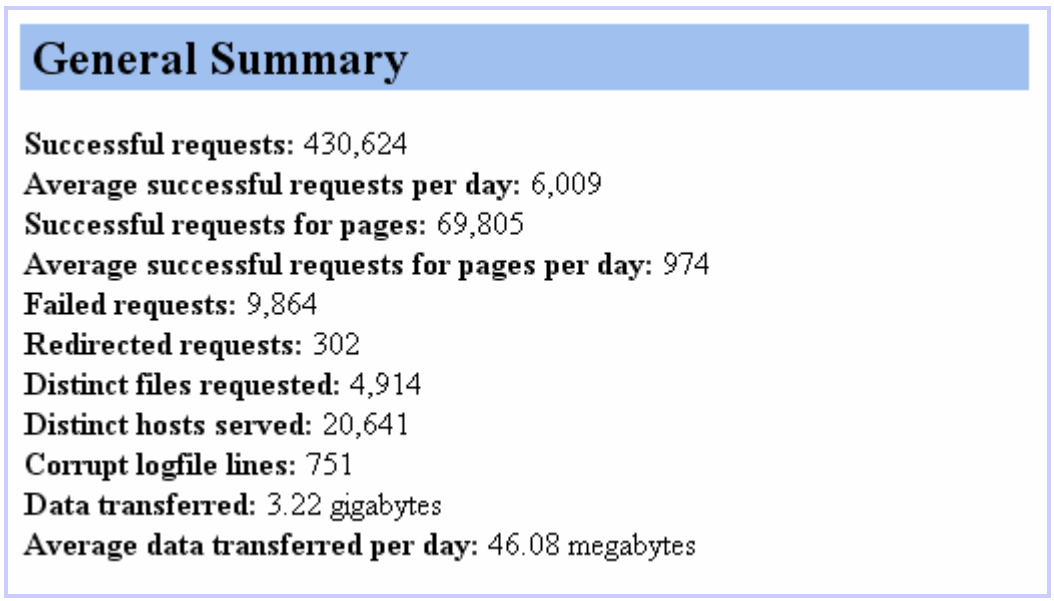
Per *richieste fallite* si intendono quelle in cui il codice di risposta HTTP è del tipo *4xx* (errore nella richiesta) o *5xx* (errore del server). Questi responsi spesso si hanno perché il file richiesto non è stato trovato o è protetto in lettura.

Le *richieste re-dirette* (inoltrate ad altro URI) sono quelle il cui codice di risposta HTTP è del tipo *3xx* ma non è *304,* cioè l'utente viene indirizzato verso un'altra risorsa. Di solito ciò avviene perché il client ha richiesto una directory senza scrivere il simbolo *backslash* '\' finale: il server, dunque, risponde indicando un diverso URI e il client formula una seconda richiesta riferendosi in modo corretto alla risorsa desiderata. Ciò capita frequentemente anche quando si cerca di seguire link di banner pubblicitari.

Sono considerate *linee corrotte* le linee il cui formato non è quello previsto. I *file* sono gli URL presenti nelle richieste: possono rappresentare pagine, oggetti inclusi nelle pagine (immagini, suoni, ecc.), directory, invocazioni di script, mentre gli *host* sono i dispositivi che inviano la richiesta di un file al server.

3.2. Descrizione delle interrogazioni effettuate

Il primo obiettivo è stato quindi riuscire a estrarre, utilizzando la struttura progettata, le informazioni fornite dai principali report, facendo riferimento, in particolare al *General Summary* generato da *Analog*, di cui viene mostrato un esempio



General Summary

Successful requests: 430,624
Average successful requests per day: 6,009
Successful requests for pages: 69,805
Average successful requests for pages per day: 974
Failed requests: 9,864
Redirected requests: 302
Distinct files requested: 4,914
Distinct hosts served: 20,641
Corrupt logfile lines: 751
Data transferred: 3.22 gigabytes
Average data transferred per day: 46.08 megabytes

Figura 2 – *General summary - Analog*

3.2.1. Richieste che hanno avuto successo

Il primo dato indicato si riferisce alle richieste che hanno avuto successo. Per ricostruire tale dato abbiamo utilizzato la seguente query (scritta, come tutte le altre, in T-SQL):

```
select @richiesteConSuccesso = count(*)

from Linea

where (statoL >= 200 and statoL < 300) or statoL = 304
```

sono state contate le sole linee il cui codice era uno di quelli che indicano successo (del tipo *2xx* oppure di valore *304*). *Analog* prevede un comando *304ISSUCCESS* che permette di specificare se includere il valore *304* tra i codici di risposta rappresentanti un successo.

3.2.2. Richieste che hanno avuto successo in media al giorno

Il secondo dato rappresenta invece le richieste che hanno avuto successo in media ogni giorno.

```
select @giorni = DATEDIFF(ss, min(dataD), max(dataD))

from Data

set @giorni = @giorni / ( 60 * 60 * 24 )

set @richiesteConSuccessoInMediaAlGiorno =

@richiesteConSuccesso/@giorni
```

Per giorno non si intende una giornata del calendario (ad es. la data 26/09/2005), bensì un intervallo costituito da ventiquattro ore, cioè da 86400 secondi (60 secondi * 60 minuti * 24 ore). Dunque si conta la quantità di intervalli di questo tipo presente nei dati considerati: tale quantità non è necessariamente intera; essa viene calcolata valutando quanti secondi sono passati tra la prima (in ordine di tempo) richiesta memorizzata e l'ultima.

Una scelta alternativa sarebbe quella di valutare questa media solo in riferimento ai giorni dei quali sono state registrate tutte le richieste. In tal caso il valore *@giorni* sarebbe un numero intero e la query risulterebbe la seguente:

```
select @giorni = distinct giornoD, meseD, annoD

from Data

set @richiesteConSuccessoInMediaAlGiorno = @richiesteConSuccesso /

@giorni
```

Tra le due alternative è preferibile la prima, in quanto è possibile ricavare le informazioni generate dalla seconda interrogazione restringendo le linee considerate ai giorni di cui si hanno tutte le richieste, mentre non è possibile, utilizzando la seconda query, ricavare le informazioni generate dalla prima. La media, in tutti i casi, è la somma delle richieste che hanno avuto successo (contante precedentemente) diviso il numero di giorni contati.

3.2.3. Richieste per pagine che hanno avuto successo

Si passa poi alle richieste con successo relative a pagine

```
select @richiesteConSuccessoPerPagine = count(*)

from Linea join Url on richiestaL = nomeU

where((statoL >= 200 and statoL < 300) or statoL = 304)

and (tipoU = 'html' or tipoU = 'htm'

or right(nomeU, 1) = '/'

or charindex('/?', nomeU, 1) > 0

or charindex('/#', nomeU, 1) > 0)
```

In questo caso sono state scelte tutte le linee che hanno avuto successo, il cui URL rappresenta una pagina.

Sono state considerate pagine tutti i file con estensione *htm* o *html*, e le directory (URL il cui ultimo carattere è il simbolo *backslash* '\'). Nel caso di URL contenenti parametri non vengono considerati tutti i caratteri successivi alla prima occorrenza del simbolo *punto interrogativo* ('?') o *cancelletto* ('#') (simboli che indicano, appunto, la presenza di parametri).

Per pagina, infatti si intende ogni schermata visualizzata dall'utente, indipendentemente dal numero di oggetti di cui essa è composta. Non è però banale distinguere tutte le pagine da considerare e a tale proposito viene fornito qualche esempio. Non è detto che un file *html* (o *htm*) sia necessariamente una pagina: essa potrebbe essere costituita da più frame ciascuno dei quali contenente un file *html*; le pagine create dinamicamente, il cui aspetto varia anche solo leggermente in base al comportamento dell'utente possono essere considerate diverse, esaminiamo ad esempio il seguente caso: due utenti visualizzano la stessa pagina di uno stesso sito il cui sfondo è bianco; dopo aver indicato il sesso (selezionando la relativa opzione) lo sfondo diventa rosa nel caso il sesso selezionato sia femminile, azzurro in caso contrario; occorre decidere se le pagine viste da ogni utente sono due oppure una. Tra le pagine si potrebbero includere quelle in formato *asp*, *jsp, cgi,* (pagine create dinamicamente), o *xml* (un altro formato che può essere utilizzato per pagine web).

Una soluzione possibile alla scelta dei file da considerare pagine è la conoscenza degli URL che costituiscono il sito. In tal caso si selezionerebbero solo gli URL conosciuti come tali o

si potrebbe aggiungere un attributo alla tabella URL indicante la tipologia del file (pagina o oggetto), ottenendo una query semplificata del tipo:

```
select @richiesteConSuccessoPerPagine = count(*)

from Linea join Url on richiestaL = nomeU

where((statoL >= 200 and statoL < 300) or statoL = 304)

and tipologiaU = 'pagina'
```

Analog, di default, considera pagine le directory e i file in formato *html* o *htm*, ma permette di includere o escludere determinati URL dal conteggio delle pagine utilizzando i comandi *PAGEINCLUDE* e *PAGEEXCLUDE*. Solitamente, inoltre, vengono esclusi dal match i parametri, ma altri due comandi, *ARGSINCLUDE* and *ARGSEXCLUDE*, consentono di modificare questa opzione.

Questa interrogazione è l'unica che porta a risultati leggermente differenti da quelli di *Analog* in quanto un URL viene considerato differente dallo stesso seguito dal *backslash*, mentre il software commerciale non effettua questa distinzione, considerando come stessa pagina, ad esempio, sai */indirizzo/*, sia */indirizzo*.

3.2.4. Richieste per pagine che hanno avuto successo in media al giorno

Per determinare il numero di richieste, relative a pagine, che hanno avuto successo in media ogni giorno, basta utilizzare le informazioni ricavate precedentemente:

```
set @richiesteConSuccessoPerPagineInMediaAlGiorno =
@richiesteConSuccessoPerPagine / @giorni
```

3.2.5. Linee corrotte

Di seguito è riportata l'interrogazione relativa le linee corrotte:

```
select @lineeCorrotte=count(*)
from Linea
where statoL >= 600
or charindex('"', metodoL) > 0
or charindex('"', richiestaL) > 0
or charindex('"', protocolloL) > 0
or right(clientL, 1) = ' ' or left(clientL, 1) = ' '
or right(rfcL, 1) = ' ' or left(rfcL, 1) = ' '
or right(accountL, 1) = ' ' or left(accountL, 1) = ' '
or right(fusoL, 1) = ' ' or left(fusoL, 1) = ' '
or right(metodoL,1 ) = ' ' or left(metodoL, 1) = ' '
or right(richiestaL, 1) = ' ' or left(richiestaL, 1) = ' '
or right(protocolloL, 1) = ' ' or left(protocolloL, 1) = ' '
or right(versioneL, 1) = ' ' or left(versioneL, 1) = ' '
or right(statoL, 1) = ' ' or left(statoL, 1) = ' '
or right(dimL, 1) = ' ' or left(dimL, 1) = ' '
```

Vengono considerate linee corrotte quelle aventi codice di risposta superiore o uguale a *600* e quelle in cui è presente uno *spazio vuoto* (‘ ‘) o sono presenti le *doppie virgolette* (‘ ” ’) all’interno di uno dei singoli campi.

In alcuni casi, però, vengono utilizzati dai programmatori anche i codici di stato di valore superiore a *600*, quindi la query dovrebbe essere adattata al caso specifico. Si potrebbe anche includere nella tabella dei codici di stato un attributo per specificare la categoria di appartenenza di tale codice (invio ad altro server, successo, fallimento, inutilizzato, ecc.): così nella clausola *from* dovrebbe essere utilizzata una operazione di *join*

```
from Linea join Stato on statoL = nomeT
```

e la condizione

```
where statoL >= 600
```

diventerebbe

```
where categoriaT = 'inutilizzato'
```

Per quanto riguarda il formato dei singoli campi occorre fare una considerazione. In questo progetto, nella fase di pre-elaborazione, occorre indicare quali campi sono presenti nella linea e quali caratteri separano campi diversi. Dunque se nella stringa formato si specifica, ad esempio, che le linee sono di questo tipo: *%h”%u*, si potrebbe intendere che tutto ciò che segue le *doppie virgolette* fa parte della stringa che identifica l’utente, quindi nel caso si abbia una linea del tipo *128.23.23.45””riky* si potrebbe considerare la linea non corrotta ma esatta e si potrebbe considerare il campo *%u* avente valore *“ricky*. Oppure si potrebbe ammettere la ripetizione dei separatori e considerare la linea corretta e il valore del campo *%u* corrispondente a *ricky*.

Nel primo caso la query cambierebbe in quanto sarebbe presente solo una condizione (*statoL >= 600*); nel secondo caso, invece, occorrerebbe operare sul valore del campo in fase di pre-elaborazione o post-elaborazione (dopo l’inserimento nel database).

3.2.6. Richieste fallite

Per contare le richieste fallite é stata formulata la query:

```
select @richiesteFallite = count(*)

from Linea

where statoL >= 400 and statoL < 600

and charindex('"', metodoL) <= 0

and charindex('"', richiestaL) <= 0

and charindex('"', protocolloL) <= 0

and right(clientL,1) <> ' ' and left(clientL,1) <> ' '

and right(rfcL,1) <> ' ' and left(rfcL,1) <> ' '

and right(accountL,1) <> ' ' and left(accountL,1) <> ' '

and right(fusoL,1) <> ' ' and left(fusoL,1) <> ' '

and right(metodoL,1) <> ' ' and left(metodoL,1) <> ' '

and right(richiestaL,1) <> ' ' and left(richiestaL,1) <> ' '

and right(ptotocolloL,1)<>' ' and left(protocolloL,1) <> ' '

and right(versioneL,1) <> ' ' and left(versioneL, 1) <> ' '

and right(statoL,1) <> ' ' and left(statoL,1) <> ' '
```

Come specificato in precedenza vengono considerate tutte le richieste il cui codice di risposta é del tipo *4xx* o *5xx*, mentre non vengono considerate le linee considerate corrotte in quanto alcuni dei campi non sono nel giusto formato. A proposito di questa scelta valgono le considerazioni fatte nel paragrafo precedente.

3.2.7. Richieste inoltrate ad altro URI

Le richieste inoltrate ad altri URI sono contate mediante la seguente interrogazione

```
select @richiesteRedirette = count(*)

from Linea

where statoL >= 300 and statoL < 400 and statoL <> 304

and charindex('"', metodoL) <= 0

and charindex('"', richiestaL) <= 0

and charindex('"', protocolloL) <= 0

and right(clientL,1) <> ' ' and left(clientL,1) <> ' '

and right(rfcL,1) <> ' ' and left(rfcL,1) <> ' '

and right(accountL,1) <> ' ' and left(accountL,1) <> ' '

and right(fusoL,1) <> ' ' and left(fusoL,1) <> ' '

and right(metodoL,1) <> ' ' and left(metodoL,1) <> ' '

and right(richiestaL,1) <> ' ' and left(richiestaL,1) <> ' '

and right(ptotocolloL,1)<>' ' and left(protocolloL,1) <> ' '

and right(versioneL,1) <> ' ' and left(versioneL,1) <> ' '
```

Anche in questo caso valgono le considerazioni fatte in precedenza relativamente al formato dei campi della linea; la query é del tutto analoga a quella effettuata per determinare le richieste fallite con la sola differenza che si considerano tutte le richieste il cui codice HTTP é del tipo *3xx* e non quelle il cui codice vale *304*.

3.2.8. File richiesti distinti

Si considerano ora i diversi file che sono stati richiesti

```
select distinct nomeU

from Url join Linea on nomeU=richiestaL

where ((statoL >=200 and statoL<300) or statoL=304)

and charindex('"',metodoL)=0 and charindex('"',richiestaL)=0

and charindex('"',protocolloL) = 0

and right(clientL,1) <> ' ' and left(clientL,1) <> ' '

and right(rfcL,1) <> ' ' and left(rfcL,1) <> ' '

and right(accountL,1) <> ' ' and left(accountL,1) <> ' '

and right(fusoL,1) <> ' ' and left(fusoL,1) <> ' '

and right(metodoL,1) <> ' ' and left(metodoL,1) <> ' '

and right(richiestaL,1) <> ' ' and left(richiestaL,1) <> ' '

and right(ptotocolloL,1)<>' ' and left(protocolloL,1) <> ' '

and right(versioneL,1) <> ' ' and left(versioneL,1) <> ' '

set @fileRichiestiDistinti=@@rowcount
```

Vengono selezionate tutte le linee che hanno avuto successo e che non sono corrotte, si potrebbe scegliere in alternativa di contare tutti gli URL presenti, indipendentemente dall'esito della richiesta, oppure si potrebbero contare solo quelli appartenenti ai siti cui si riferiscono le linee caricate (nel caso si abbia tale dato); dopodiché si contano i diversi URL presenti in tale raccolta, indipendentemente dalla loro estensione e dal loro formato. Nel caso più richieste contenenti parametri diversi si riferiscano allo stesso URL vengono contate più volte, ad esempio, gli URL */cgi-bin/script.pl?x=1&y=2* e */cgi-bin/script.pl?x=5&y=7* sono considerati file differenti in quanto generano effetti diversi.

3.2.9. Host serviti distinti

Per host si intende l'entità che effettua la richiesta.

```
select distinct ipC

from Client join Linea on ipC=clientL

where (statoL>=200 and statoL<300) or statoL=304
```

Vengono considerati tutti gli host che hanno effettuato almeno una richiesta che ha avuto successo. Ogni host é identificato dall'indirizzo IP e rappresenta, dunque, il dispositivo che ha effettuato la richiesta ma non l'utente; potrebbe accadere, infatti, che diversi utenti utilizzino, magari in tempi differenti, la stessa macchina; bisogna inoltre tener conto che dei casi in cui viene utilizzato un proxy server (vedi cap. 5, sez. 6).

3.2.10. Dati trasferiti

Viene considerata ora la quantità di byte trasferiti per ogni richiesta registrata

```
select @datiTrasferiti=sum((dimL as bigint))

from Linea

where dimL<>'-'

and ((statoL>=200 and statoL<300) or statoL=304)
```

Nella query vengono selezionate tutte le linee che hanno avuto successo e si sommano i valori *dimL* di ognuna, a meno che non siano nulli ('-'). *dimL* rappresenta la dimensione del corpo della risposta inviata all'utente che ha effettuato la richiesta. Tale dimensione non sempre rappresenta l'intera dimensione del file, talvolta, infatti, viene inviata all'host solo una parte del documento e in seguito, se richiesta, verrà inviata la parte restante. Dunque questa interrogazione rappresenta la dimensione dei dati inviati dal server verso l'host e non la dimensione dei file richiesti.

L'informazione ricavata non rappresenta neanche la dimensione dei documenti (del sito visitato) ricevuti dall'utente: i dati potrebbero essere inviati dal server verso l'host che ne ha fatto richiesta ma non arrivare a causa di mal funzionamenti della rete o a causa della chiusura della comunicazione da parte dell'host stesso; l'host potrebbe aver memorizzato in una cache (ad esempio quella del browser) parte delle informazioni contenute dei documenti del sito visitato:

nel caso di reti che utilizzano *firewall*, per ridurre il carico di rete, il proxy server agisce come una macchina di copia locale. Quando una pagina è caricata in un browser attraverso un proxy server, il proxy salva una copia di questa pagina nella sua cache. In questo modo, un documento che viene richiesto molto spesso dagli utenti della stessa rete locale, ha bisogno di essere trasferito al proxy solo una volta, che poi risponderà alle future richieste delle stessa pagina dalla sua cache locale invece che connettersi al server web da cui il documento è originariamente stato tratto.

3.3. Altri report

Oltre al *General Summary,* sono stati realizzati, utilizzando il database progettato, altri report di *Analog*. Sono stati realizzati resoconti dedicati a singoli ambiti (ad esempio dimensione del corpo delle risposte) che indicano, per ogni categoria dell'ambito a cui ci si riferisce, la quantità di richieste che hanno avuto successo ed eventualmente altre informazioni (es. quante richieste fanno riferimento a pagine).

Esaminiamo il *resoconto dei tipi di file richiesti*. In questo caso lo scopo è elencare, per ogni tipo di file, il numero di richieste che facevano riferimento a una risorsa di quel tipo e la percentuale di byte appartenenti a documenti di tale tipo trasferiti.

3.3.1. Tabelle temporanee

Per raccogliere i dati è stata utilizzata una tabella temporanea locale di ausilio. È stata effettuata questa scelta in quanto utilizzare una tabella facilita l'implementazione dell'interrogazione trattata e in quanto dati non devono persistere nel tempo, ma devono essere solo visualizzati, e le tabelle locali sono visibili solo nella sessione corrente. Ogni riga della tabella deve contenere i tre valori da visualizzare: tipo di file, richieste con successo, percentuale di byte trasferiti. Ecco dunque l'istruzione per la creazione:

```
create table #temp (tipo varchar(10),richieste int, percentuale
float)
```

Dopodiché nella prima colonna sono stati inseriti i valori relativi alle estensioni di file, considerando solo quelle rappresentanti il formato di almeno una delle risorse richieste:

```
INSERT #temp(tipo) (

       select distinct tipoU

       from Url join Linea on nomeU=richiestaL

       where tipoU <> '-')
```

È stato poi creato un cursore per scorrere una ad una le righe di questa tabella e popolarne le altre colonne. Per ogni valore di *tipo di file* presente in tabella sono stati contati i byte trasferiti in risposta alle richieste per risorse di quel tipo e sono state contate le richieste con successo.

```
select tipoU, metodoL, richiestaL, protocolloL
```

```
from Linea join Url on richiestaL=nomeU

where tipoU=@tipo

group by tipoU,metodoL,richiestaL,protocolloL

set @r=@@rowcount

update #temp

set richieste=@r

where tipo=@tipo

select @tot=count(*)

from Url

where tipoU=@tipo

update #temp

set percentuale=@tot/@totale

where tipo=@tipo
```

La variabile @totale è il numero di richieste con successo totali.

Finiti gli inserimenti è bastato visualizzare tutta la tabella per ottenere il resoconto desiderato

	Tipo	Richieste	Percentuale
1	html	11	0.33333333333333331
2	pdf	6	0.18181818181818182
3	gif	4	0.12121212121212122
4	jpg	2	6.0606060606060608E-2
5	ppt	1	3.0303030303030304E-2
6	txt	1	3.0303030303030304E-2

Figura 3 – Report *Tipi di File - Query Analyzer*

4. Sessioni e percorsi

Definire esattamente il concetto di sessione non è immediato. In generale per sessione si intende un insieme di visite effettuate da un utente rilevate e tracciate dal web server, cioè l'itinerario effettuato da un singolo utente in un certo arco di tempo[22]. La difficoltà sta nell'identificare gli utenti, nel definire cos'è una visita e nello scegliere un arco di tempo adeguato.

4.1. Considerazioni preliminari

4.1.1. Utente

Per utente si intende la persona umana che *visita* i siti web. Tra le informazioni presenti nel file di log, per riconoscere un utente, possono essere utilizzati essenzialmente l'indirizzo IP e l'autenticazione fornita dall'utente stesso.

L'indirizzo IP permette di distinguere il dispositivo utilizzato per navigare, ma da solo non consente di identificare la persona in quanto individui diversi possono utilizzare la stessa macchina (di solito in tempi diversi). Occorre poi tener conto che, nel caso di proxy, dispositivi differenti vengono rappresentati dallo stesso indirizzo (quello del proxy stesso) e che solo una parte degli indirizzi IP assegnati sono statici, mentre gli altri sono assegnati dinamicamente ogni qualvolta l'utente si colleghi ad Internet. Dunque l'indirizzo IP identifica univocamente un dispositivo in un intervallo di tempo più o meno ampio.

Per quanto riguarda l'autenticazione dell'utente essa è un nome scelto dallo stesso per farsi riconoscere. Occorre tener conto, però che la stessa autenticazione può essere utilizzata da più individui: da persone appartenenti a uno stesso gruppo o da persone che in qualche modo conoscono uno username per autenticarsi pur non essendo titolari dell'account utilizzato. Può capitare, ad esempio, che un professore fornisca ai propri studenti un account unico per accedere al materiale didattico online; oppure può succedere che un individuo acceda alle risorse di un sito utilizzando l'account di un parente per non eseguire tutta la procedura di iscrizione.

[22] http://www.pc-facile.com/glossario/sessione/

4.1.2. Visita

Una vista è una sequenza di richieste consecutive effettuate da uno stesso visitatore allo stesso sito. La visita inizia con il primo hit, che può registrare anche la provenienza dell'utente (nel caso venga indicato il *referrer*), e termina dopo la scadenza del timeout dall'ultimo hit registrato, cioè dopo una durata massima. Non esiste uno standard per questa durata e neppure un consistente accordo in proposito. La lunghezza di una sessione può variare da un minimo di dieci/quindici minuti ad un massimo di un'ora. Nella maggior parte dei casi essa è impostata su venti o trenta minuti. Se dura venti minuti, ciò significa che ad un utente unico vengono attribuite due visite al sito, nel caso in cui una sua richiesta di pagina giunga oltre venti minuti dopo la precedente richiesta registrata. Viceversa, se l'intervallo trascorso tra questi due eventi è inferiore a venti minuti, allora viene conteggiata per quell'utente un'unica visita. Come è facile comprendere, la durata di sessione è un parametro del tutto arbitrario, che nulla ha a che vedere con l'effettivo comportamento degli utenti collegati ad un sito e che può tuttavia influenzare le valutazioni del settore commerciale di un'azienda, circa la misura della fedeltà degli utenti ai siti presi in considerazione. Poniamo ad esempio che un sito, avendo un timeout di sessione impostato su venti minuti, registri molte visite di utenti unici nell'arco di un mese di rilevazione: se ne potrebbe ricavare l'idea che dietro quelle visite ripetute si celino utenti fidelizzati. Basterebbe però probabilmente aumentare di soli dieci minuti il timeout di sessione, per scoprire che il numero di visite al sito da parte di utenti unici è nettamente diminuito. È dunque la conoscenza del significato e della reale portata dei numeri offerti dalle statistiche di traffico che aiuta a non commettere pericolosi errori di valutazione[23].

[23] http://www.regione.emilia-romagna.it/sin_info/lineeguida/12_statistiche_glossario_termini.htm

4.2. Definizione di una sessione

Come prima approssimazione la scelta è identificare gli utenti con la combinazione dei campi del log file *host auth_user*. Si suppone cioè che su una macchina lavori un utente alla volta e che lo username sia personale (e che nessuno utilizzi quello di un altro individuo).

A proposito di ogni utente si vuole determinare quali pagine ha visitato (per la definizione di *pagina* si faccia riferimento al paragrafo *Richieste per pagine che hanno avuto successo*, cap. 3, sez. 2.1). Per quanto riguarda il timeout da considerare si rimanda la scelta a lavori futuri, per il momento non si stabilisce alcun limite temporale delle sessioni.

4.2.1. Viste

Per determinare le sessioni sono state create delle viste SQL. Le viste possono essere immaginate come tabelle. Quando viene creata una vista nel database è archiviata l'istruzione *select* utilizzata per la creazione stessa. Il set di risultati di tale istruzione compone la tabella virtuale restituita dalla vista: esso non viene in genere salvato nel database, ma viene incorporato in modo dinamico nella logica dell'istruzione e viene generato in modo dinamico in fase di esecuzione. È possibile utilizzare questa tabella virtuale facendo riferimento al nome della vista nelle istruzioni *Transact-SQL*, con la stessa modalità utilizzata per fare riferimento a una tabella.

Per determinare le sessioni occorre estrarre sottoinsiemi di dati registrati nel database e gestirli in formato tabellare (SQL), ma non occorre replicarli essendo essi già memorizzati nelle tabelle; la replicazione, infatti, sarebbe piuttosto onerosa in quanto, come già detto in precedenza, in questo ambito si gestiscono grandi moli di dati; per questo motivo risulta conveniente utilizzare le viste.

Sono state create delle viste *base* sulle quali operare per ottenere poi i risultati veri e propri. La vista *suc_req* seleziona le sole linee che hanno avuto successo (questa vista può essere utilizzata anche per migliorare l'implementazione del *General Summary* e degli altri report)

```
create view suc_req
(idL,clientL,rfcL,accountL,dataL,fusoL,metodoL,richiestaL,protocollo
L,versioneL,statoL,dimL,fileL,serverL,errL)

as

select *

from Linea

where (statoL>=200 and statoL<300) or statoL=304
```

e in modo analogo le viste *pagine* e *metodi_utente* selezionano la prima gli indirizzi di pagine web presenti nella tabella *Url* (anche in questo caso si veda il paragrafo *Richieste per pagine che hanno avuto successo*, cap. 3, sez. 2.1) e la seconda i metodi utilizzati dalle persone per visualizzare informazioni del web (vedi *Profili utenti*, cap. 5, sez. 6).

Utilizzando le viste appena esaminate si può creare facilmente una vista interessante rappresentante tutti i passi effettuati da ogni navigatore che si sia autenticato

```
create view passi_utente (ip, account, url, data, idL)

as

select distinct clientL, accountL, nomeU, dataL, idL

from Client join suc_req on ipC=clientL join pagine on
nomeU=richiestaL join metodi_utente on nomeM=metodoL

where accountL<>'-'
```

Sfruttando questa vista ne è stata creata un'altra, *sessione_0_1*, che seleziona, per ogni utente, solo la prima e l'ultima risorsa richiesta (con la relativa data). Lavorando sulle ultime due viste descritte, e utilizzando un cursore è stata creata la prima rappresentazione di sessione che per ogni utente descrive la prima e l'ultima pagina richieste e il numero di pagine a cui ha avuto accesso.

In seguito si è passati ad esaminare i percorsi degli utenti. In particolare è stata creata una vista che raccogliesse, definiti due nodi (due URL di pagine), tutte le sessioni contenenti ordinatamente almeno una richiesta per il primo nodo e almeno una richiesta per il secondo. Poi è stata creata un'ulteriore vista per selezionare, dalla vista precedente, solo le sessioni in cui i due nodi definiti fossero adiacenti (cioè richiesti uno di seguito all'altro).

4.2.2. Rappresentazione grafica

I risultati ottenuti sono visualizzati in ambiente *Query Analyzer* in formato tabellare. Con un programma *Perl* tali dati sono stati esportati in un file di testo e, elaborando tale file ne è stato creato un altro leggibile dal software open source *Graphviz*[24]. *Graph Visualization* (questo il nome esteso) permette di rappresentare informazioni strutturate sottoforma di grafi e reti. Occorre indicare la descrizione dei grafi in linguaggio testuale, specificando le caratteristiche dei nodi (nome, colore e forma col quale visualizzarlo, ecc.) e gli archi esistenti tra i vari nodi e il programma si occupa di rappresentare graficamente tale grafo creando un file nel formato specificato (es. PDF, PS, GXL, XML).

Le sessioni sono state dunque rappresentate mediante immagini del tipo

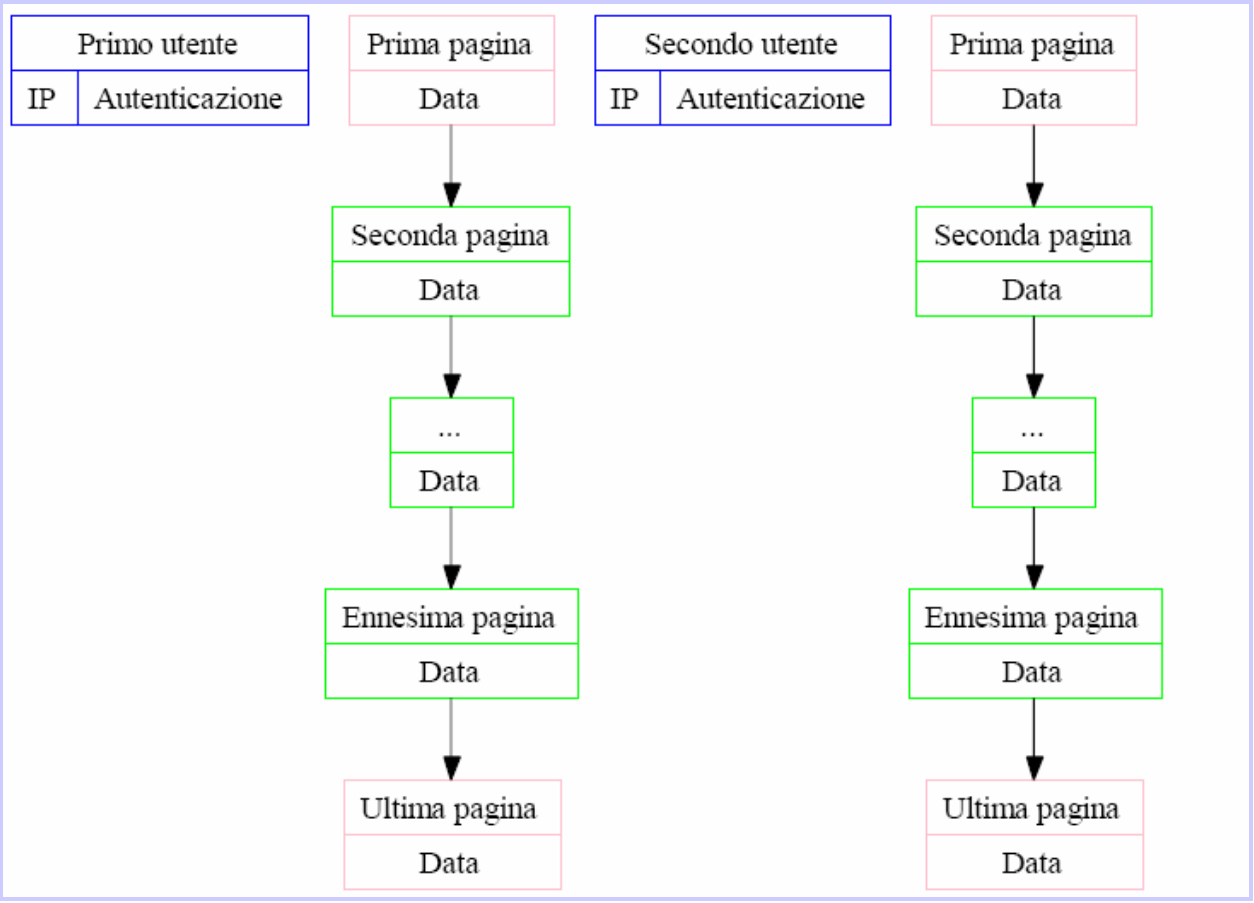


Figura 4 – Presentazione grafica di una sessione generica - *Graphviz*

[24] http://www.graphviz.org/

4.3. Percorsi

Un percorso è un insieme ordinato di URL che rappresenta l'ordine in cui sono state visitate le pagine del sito dagli utenti. Analizzando le sessioni è possibile determinare il percorso dal punto di vista di ciascun utente, ma risulta utile analizzare l'ordine in cui avvengono gli accessi al sito indipendentemente dall'identità di chi effettua le richieste. Si è scelto dunque di definire un nodo e determinare tutti i percorsi, seguiti da almeno un utente, passanti per tale nodo.

Anche in questo caso sono state utilizzate le viste è inizialmente si è cercato di ottenere i risultati proposti dagli strumenti software più diffusi. Molti di questi permettono di determinare, per ogni pagina del sito, le pagine richieste successivamente e precedentemente la visita di tale pagina, ottenendo risultati del genere:

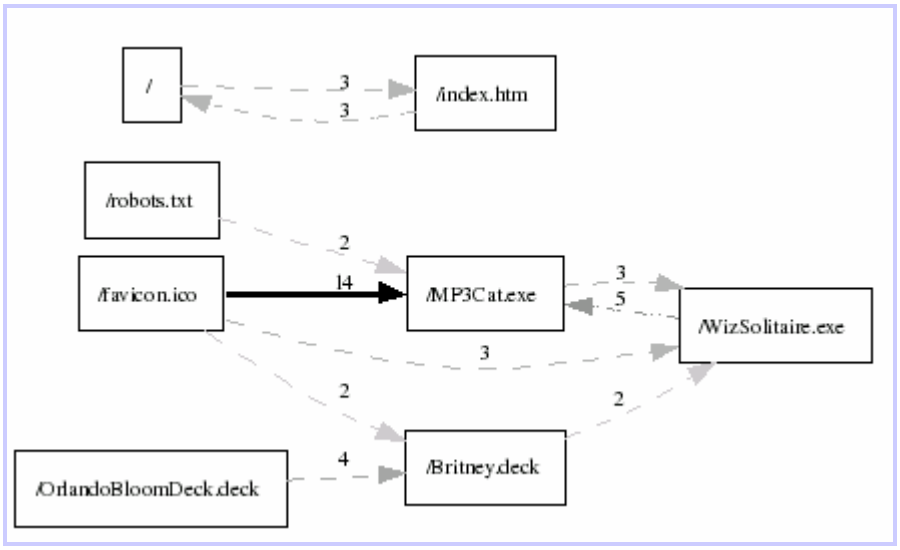


Figura 5 – Path - *LogMiner*

Nell'immagine (fornita da LogMiner) si evince che prima di visitare la pagina */Britney.deck* in quattro casi è stata richiesta la risorsa */OrlandoBloomDeck*.deck e in due casi il file */favicon.ico*, mentre in seguito alla pagina considerata *(/Britney.deck)* è stata richiesto il file */WizSolitaire.exe*.

Utilizzando le viste create in precedenza è stato invece possibile ottenere informazioni più raffinate: scelte due pagine A e B si determinano tutte le sessioni in cui è stata visitata in un certo istante la pagina A e in un istante successivo la pagina B.

È stato poi aggiunto un vincolo che implicasse l'assenza di richieste di pagine tra la visita della pagina A e quella della pagina B. I dati sono stati raccolti in viste e tabelle temporanee quindi si potranno determinare facilmente (utilizzando l'istruzione SQL *count (*)*) le frequenze degli accessi successivi e precedenti alle pagine indicate, anche se i tempi di elaborazione saranno maggiori. Un esempio è il seguente schema realizzato definendo come pagina A l'URL */bdati/labdb/MatDid.html*, e come pagina B */bdati/labdb/0304/MatDid.html*

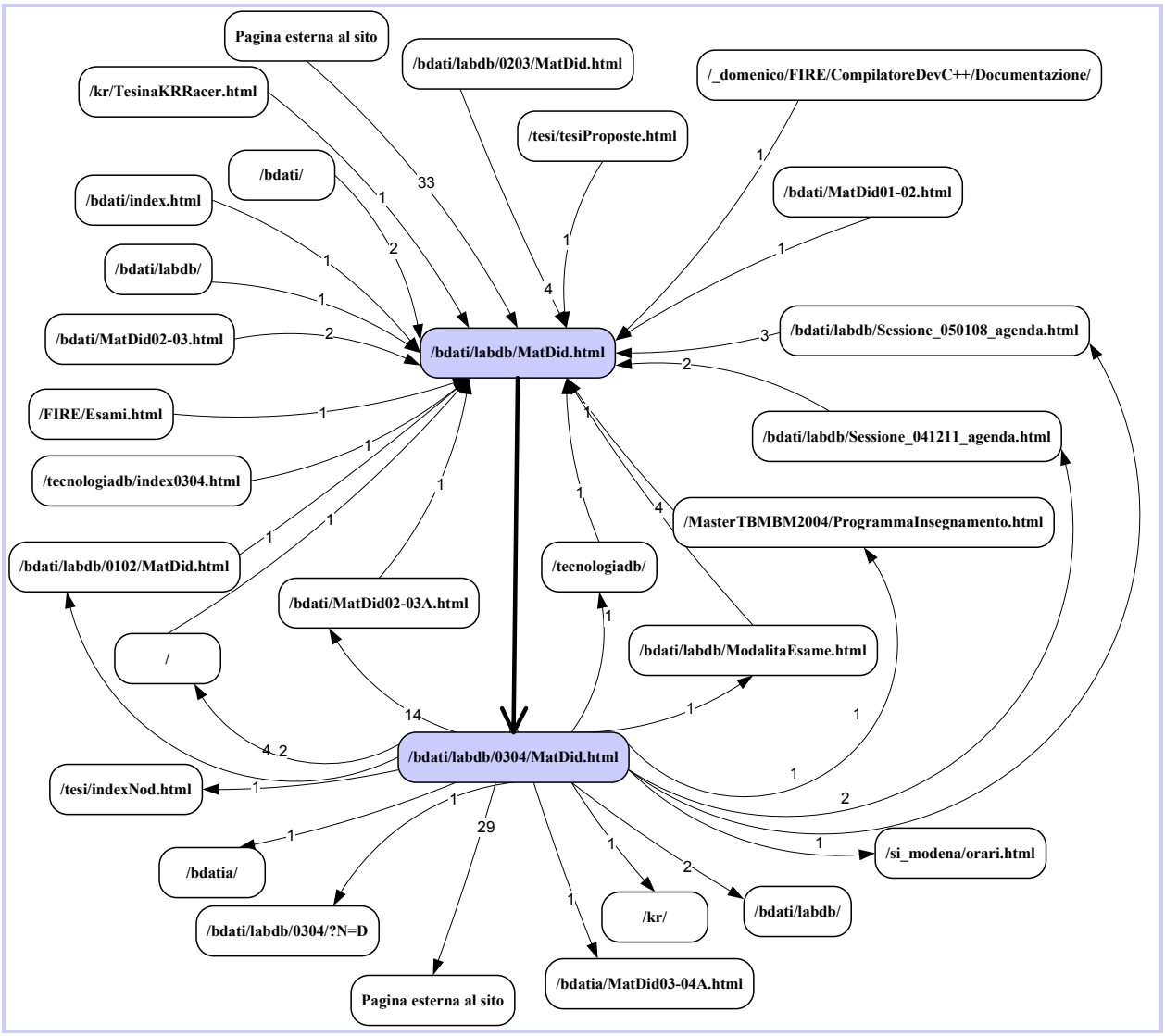


Figura 6 – Percorso passante per due pagine

5. Conclusioni e lavoro futuro

5.1. Obiettivi raggiunti

Sviluppando questa tesi sono stati raggiunti i seguenti obiettivi:

- Struttura database efficiente per raccolta e analisi dei dati dei file di log di web server;
- Creazione resoconto generale dei dati;
- Creazione di resoconti specifici ;
- Determinazione azione di sessioni ;
- Determinazione di percorsi;
- Flessibilità della struttura per utilizzi futuri;
- Tempi di elaborazione ragionevoli.

Per progettare e definire la struttura del database sono stati creati piccoli file di log contenenti poche decine o centinaia di linee. Questi file erano strutturati appositamente per verificare il corretto funzionamento della struttura e per verificarne la risposta a casi anomali o particolari. Il progetto ha un comportamento corretto rispetto ai casi (standard e anomali) presentati nei capitoli precedenti.

La struttura è stata testata anche con un file di log contenente 898000 linee avente dimensione pari a 85 megabyte. Elaborando i dati contenuti in questo file sono stai ricostruiti alcuni resoconti proposti da *Analog* ottenendo dati pressoché identici (vedi cap. 3).

I tempi di elaborazione sono stati esaminati fase per fase e di seguito sono presentati quelli relativi all'ultimo test effettuato sul file precedentemente utilizzando un computer di medie prestazioni[25]

Programma/fase	Tempo di esecuzione
formatoLog	11 minuti e 32 secondi
elaboraLog	4 minuti e 16 secondi
classiLog	5 minuti e 1 secondo
Totale pre-elaborazione	20 minuti e 49 secondi
Creazione database	2 secondi
Caricamento dati	13 minuti e 11 secondi
Post-elaborazione	43 secondi
Creazione report generale	7 minuti e 55 secondi
Creazione report formati	12 minuti e 33 secondi
Determinazione sessioni	8 ore e 5 minuti
Determinazione percorsi	52 minuti e 21 secondi

Tabella 35 – Tempi di elaborazione

Le fasi di elaborazione più lunghe risultano essere quelle relative alla determinazione di sessioni e percorsi. Ciò è dovuto principalmente alla complessità dell'analisi effettuata ma si potrebbero studiare algoritmi migliori per ottimizzare i tempi.

Il caricamento appare oneroso soprattutto se confrontato ai tempi di strumenti di analisi che non prevedono tale fase (come *Analog*), ma non sono eccessivi se si considera l'utilizzo di un database relazionale. L'utilizzo di quest'ultimo, in compenso, permette di effettuare analisi più complesse.

[25] Processore Intel Pentium III, 598 MHz, 128 MB di RAM, S.O. Microsoft Windows XP Professional versione 2002, Service Pack 2.

Uno dei risultati più significativi raggiunti è la determinazione di tutti i percorsi che includono la richiesta consecutiva e ordinata di due pagine; ora è possibile ottenere risultati del tipo[26]:

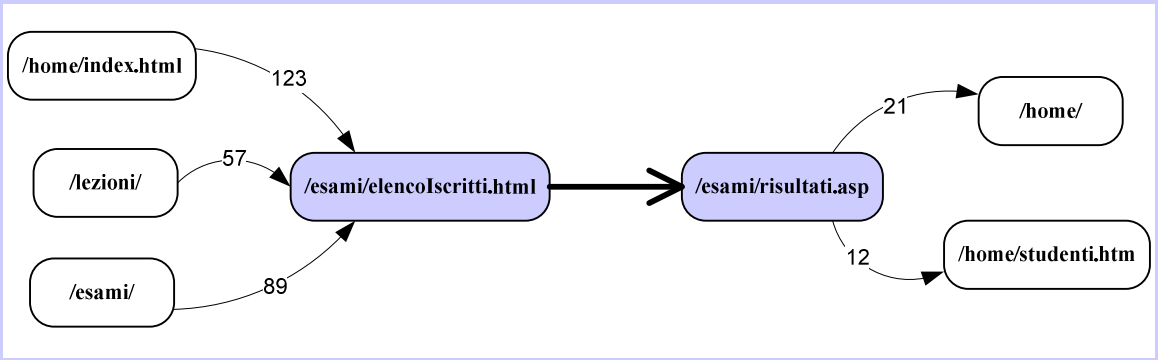


Figura 7 – Percorso passante per due pagine consecutive

Altro risultato significativo è la flessibilità del progetto. Tutte le linee del file sono state caricate nel database, il che indica che non vi è stata alcuna perdita di dati, dunque la struttura risulta flessibile a anomalie del file di log.

Soprattutto, però, effettuare nuove analisi risulta semplice e porta a tempi di definizione delle interrogazioni molto brevi: il lavoro per determinare sessioni e percorsi è stato effettuato in circa tre ore e sarebbe possibile implementare analisi più complesse, come quella relativa a percorsi passanti per nodi costituiti da più di due pagine (sez. 5 di questo capitolo).

[26] Nell'esempio ipotetico (non realizzato) sono state specificate in input le pagine */esami/elencoIscritti.html* (pagina A) e */esami/risultati.asp* (pagina B).

5.2. Presentazione dei risultati

I risultati delle interrogazioni effettuate fino adesso sono visualizzati nell'ambiente *Query Analyzer*. In futuro, però, si potrebbero rappresentare in maniera diversa. Si potrebbero, ad esempio, creare pagine web.

Nell'ambito di questo progetto occorre eseguire, per ogni resoconto che si vuole visualizzare, la relative istruzioni. Nel caso del *General Summary* si ottiene una schermata del tipo rappresentato:

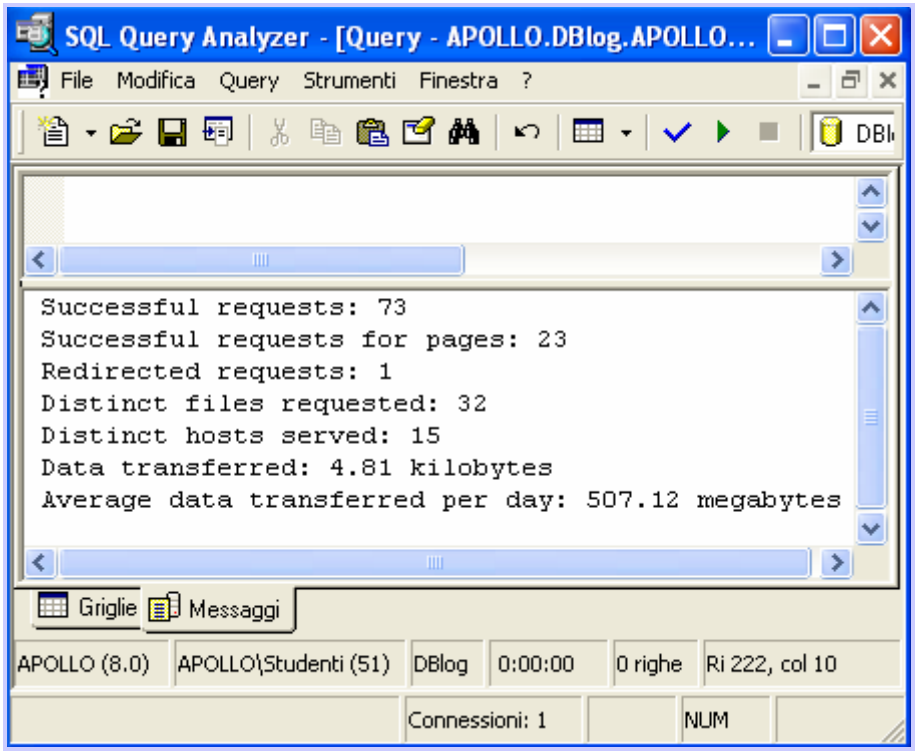
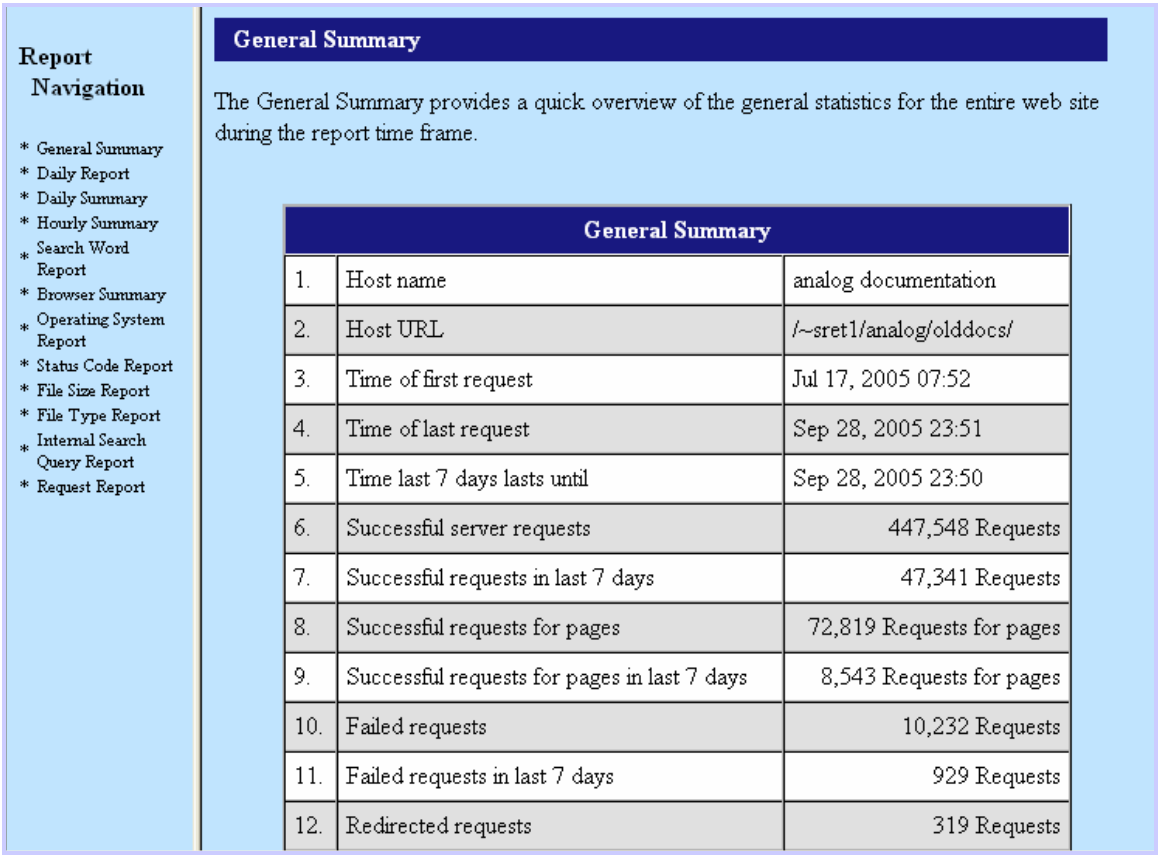


Figura 8– Report *General Summary – Query Analyzer*

Report Magic mette invece a disposizione un frame per la navigazione attraverso i report disponibili e rappresenta i risultati delle informazioni in modo più carino



Report Navigation

* General Summary
* Daily Report
* Daily Summary
* Hourly Summary
* Search Word Report
* Browser Summary
* Operating System Report
* Status Code Report
* File Size Report
* File Type Report
* Internal Search Query Report
* Request Report

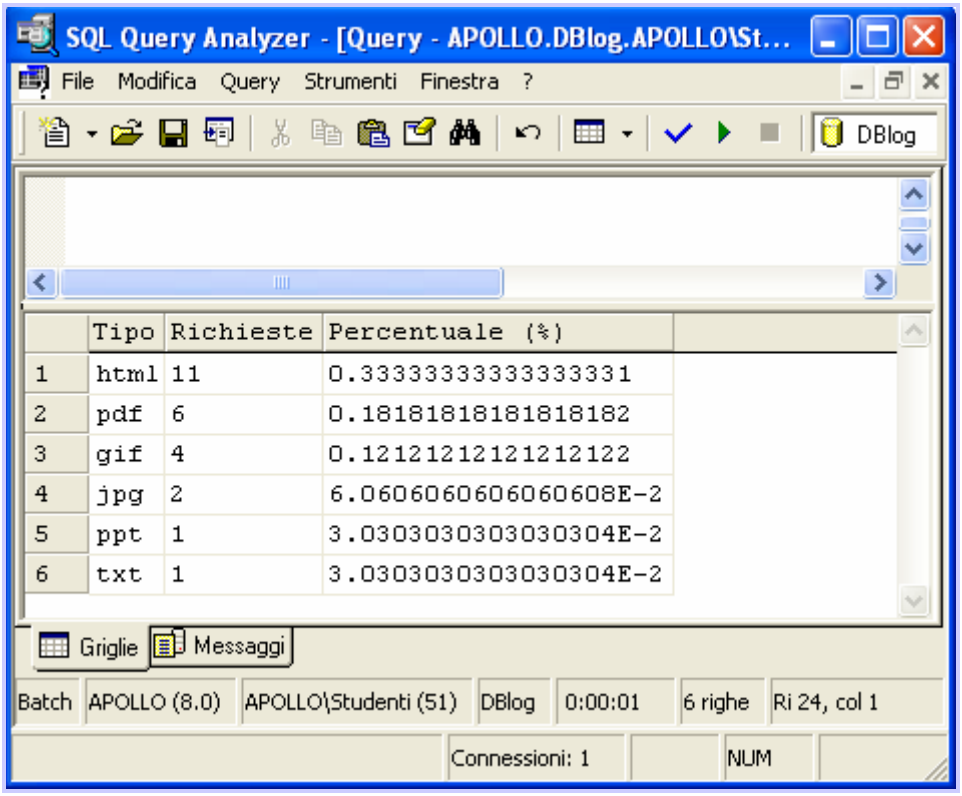
General Summary

The General Summary provides a quick overview of the general statistics for the entire web site during the report time frame.

General Summary		
1.	Host name	analog documentation
2.	Host URL	/~sret1/analog/olddocs/
3.	Time of first request	Jul 17, 2005 07:52
4.	Time of last request	Sep 28, 2005 23:51
5.	Time last 7 days lasts until	Sep 28, 2005 23:50
6.	Successful server requests	447,548 Requests
7.	Successful requests in last 7 days	47,341 Requests
8.	Successful requests for pages	72,819 Requests for pages
9.	Successful requests for pages in last 7 days	8,543 Requests for pages
10.	Failed requests	10,232 Requests
11.	Failed requests in last 7 days	929 Requests
12.	Redirected requests	319 Requests

Figura 9– Report *General Summary –Report Magic*

I risultati dell'interrogazione che permette di classificare le richieste in base il tipo di file a cui si fa riferimento vengono mostrati nell'immagine:



	Tipo	Richieste	Percentuale (%)
1	html	11	0.33333333333333331
2	pdf	6	0.18181818181818182
3	gif	4	0.12121212121212122
4	jpg	2	6.0606060606060608E-2
5	ppt	1	3.0303030303030304E-2
6	txt	1	3.0303030303030304E-2

Figura 10 – Report *Tipi di File - Query Analyzer*

Anche in questo caso i dati potrebbero essere letti da un browser e rappresentati non solo in forma tabellare

	File Type	Number of requests	Percentage of the bytes
1.	.png [PNG graphics]	281,735	14.47%
2.	.pl [Perl scripts]	65,139	41.49%
3.	.html [Hypertext Markup Language]	42,426	6.50%
4.	[directories]	30,393	36.59%
5.	.gif [GIF graphics]	23,303	0.92%
6.	.css [Cascading Style Sheets]	3,588	0.1%
7.	.ico [Windows icons]	893	0.2%
	[not listed: 6]	71	0%

Figura 11 - Report *Tipi di File - Report Magic*

ma anche in formato grafico mediante un istogramma

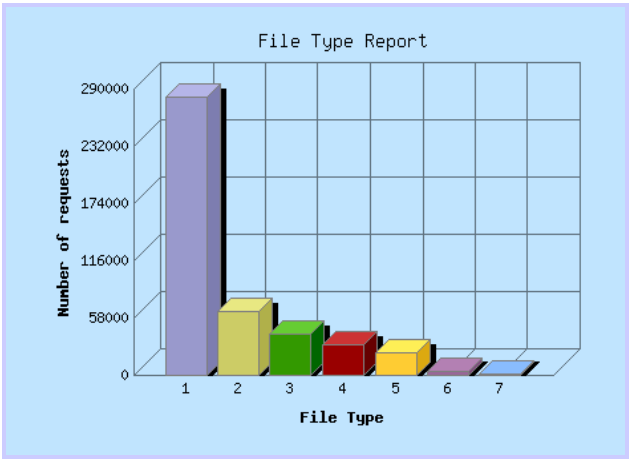


Figura 12 - Istogramma - *Report Magic*

oppure mediante un diagramma a torta

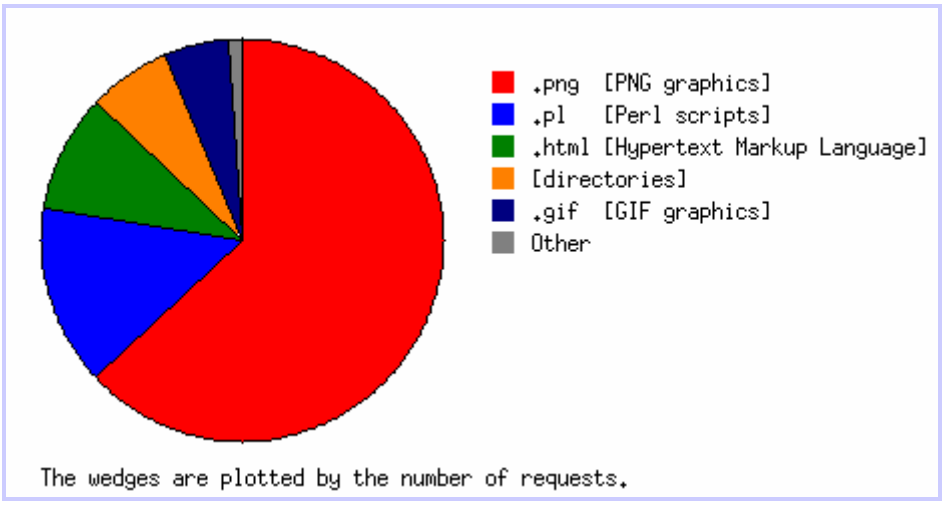


Figura 13 – Diagramma a torta – *Analog*

Si potrebbero fornire all'utente ulteriori opportunità grafiche per interagire con lo strumento creato: si potrebbe permettere la selezione dell'intervallo temporale da analizzare. *WebTrends* fornisce questa possibilità visualizzando un piccolo calendario tramite il quale è possibile selezionare il periodo scelto:

Figura 14 – Calendario – *WebTrends*

Si potrebbero rappresentare i percorsi maggiormente seguiti in modo analogo a quello utilizzato da *WebTrends*

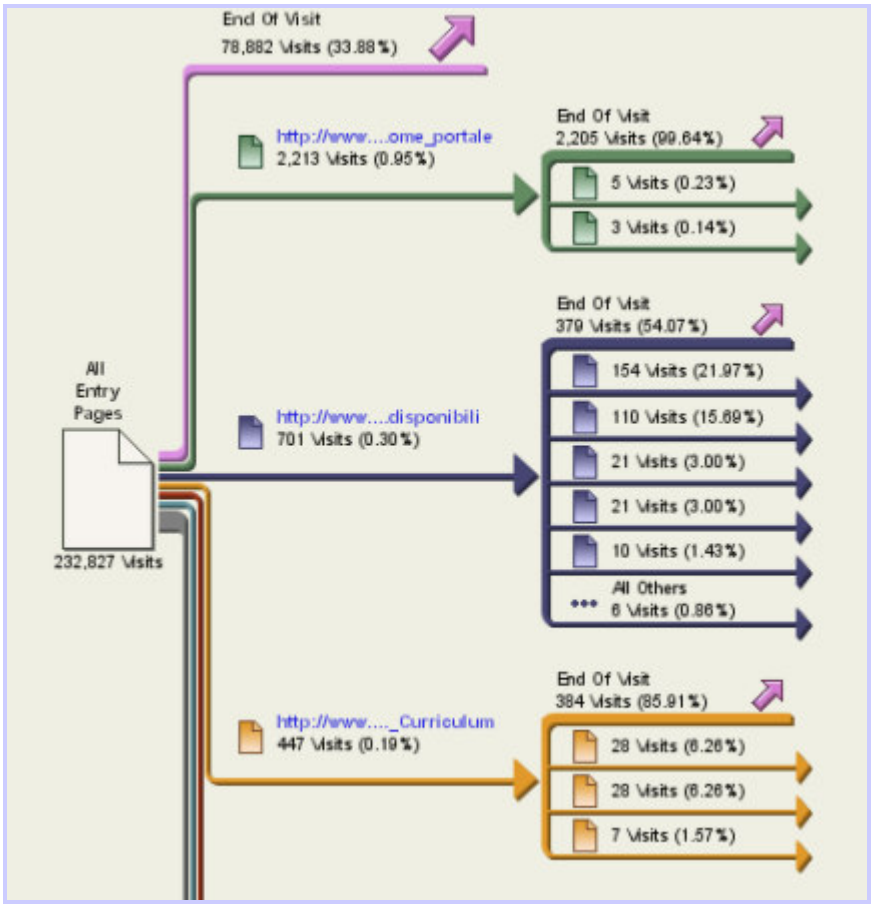


Figura 15 – Percorsi – *WebTrends*

rappresentando una pagina scelta come radice e i percorsi che sono stati seguiti a partire da tale pagina; oppure rappresentare un nucleo formato da due nodi fissati dai quali si articolano i percorsi seguiti prima di arrivare o dopo il raggiungimento di tali pagine. Per quanto riguarda le sessioni si potrebbero, per ogni utente, indicare in modo sequenziale le pagine visitate in ordine di accesso. Per questi scopi si potrebbe utilizzare il software *Graphviz* che consente di generare grafi indicando le proprietà di nodi e archi, o altri software grafici.

5.3. Gestione del formato Extended Log File Format e altri formati

Il progetto presentato prevede l'analisi di file di log di web server in formato *Common Log File Format*, ma sarebbe facilmente modificabile per gestire altri formati: sia quelli relativi ai log di server web, sia quelli relativi ai log di server FTP, firewall, proxy e mail.

Il file di log viene prima di tutto processato da un programma (*formatoLog*, vedi cap. 2, sez. 1.2) che, data una stringa rappresentante la struttura delle linee del file, si occupa di trovare i campi indicati nel parametro e riordinarli secondo un determinato schema. In questo programma è già previsto l'utilizzo di file in *Extended Log File Format*, mentre se si volesse prevedere l'utilizzo di altri formati occorrerebbe modificare parte del codice. In particolare bisognerebbe prevedere tra i campi previsti in input quelli del formato "nuovo" non presenti nei formati gia considerati e sarebbe necessario definire la posizione di questi campi nello schema che ne definisce l'ordine in output.

Il secondo programma che elabora i dati, *elaboraLog* (cap. 2, sez. 1.3) si occupa di conformarli ai formati previsti nelle fasi successive. Esso legge i valori dei campi supponendo che siano organizzati secondo l'ordine definito. Nel caso i nuovi campi siano stati inseriti in coda allo schema di ordinamento basterebbe effettuare la lettura dell'ultima porzione di linea per determinarne i valori; in caso contrario bisognerebbe adattare la lettura delle linee al nuovo schema. Sistemata la lettura si potrebbe, eventualmente, inserire la manipolazione dei nuovi campi per adattarli ai formati coi quali lavorare nelle fasi successive.

Il terzo programma di pre-elaborazione, *classiLog* (cap. 2, sez. 1.4), si occupa di leggere i dati analogamente al programma *elaboraLog*, dunque le modifiche relative a questa fase sarebbero quelle già descritte, e di riscrivere ogni campo nel file relativo, dunque basterebbe indicare i file nei quali registrare i campi aggiunti e scrivervi effettivamente i valori.

Nel database occorrerebbe creare nuove tabelle, una per ogni campo aggiunto, e prevedere le istruzioni di inserimento (analoghe a quelle già utilizzate per le altre tabelle) per popolare ciascuna caricando i dati dai file definiti nel programma *classiLog;* sarebbe anche necessario introdurre nell'entità *Linea* i riferimenti alle entità create, altrimenti si perderebbe ogni collegamento tra i nuovi valori previsti e le linee a cui appartengono.

Le interrogazioni dovrebbero essere aggiornate inserendo nuovo codice senza necessariamente modificare quello già esistente; nel caso non si volesse effettuare alcun aggiornamento esse genererebbero gli stessi risultati, ignorando le entità introdotte; non ci sarebbero inconvenienti nel creare nuove interrogazioni, in quanto non modificano i dati né la struttura del database..

5.4. Sessioni

La definizione di *sessione* scelta per ottenere i primi risultati è molto semplificata. Sarebbe necessario raffinare tale definizione e di conseguenza adattare il codice che genera i risultati. Tra i vari argomenti che si possono considerare sono rilevanti la scelta della durata delle sessione, la distinzione dei profili di utenti, la definizione di pagina.

Sono stati effettuati studi, soprattutto empirici, a proposito del tempo massimo che intercorre, durante un'unica sessione, tra una richiesta e un'altra. Questo valore varia tra i dieci e i venti minuti, dunque si potrebbero considerare appartenenti alla stessa sessione tutte le richieste successive, ricevute dal server, distanti dalla precedente un tempo considerato accettabile.

Per quanto riguarda i profili utenti si rimanda alla sezione successiva mentre relativamente le pagine si potrebbe ampliare la selezione effettuata fino ad ora. Per adesso sono state identificate come pagine documenti aventi un formato fra quelli specificati. Potrebbe essere inserita nel database una tabella contenente tutte le risorse dei siti web a cui si fa riferimento considerate pagine. In tal modo basterebbe effettuare un confronto tra gli URL presenti in tale tabella e quelli presenti nell'entità *Url* per definire quali risorse sono pagine. Se non fosse possibile avere un elenco del genere si potrebbe effettuare un'analisi della successione delle richieste ricevute: in caso di frequenti ripetizioni di successioni di richieste identiche si potrebbe pensare che ognuna di queste successioni rappresenti le richieste della pagina e degli oggetti che la compongono.

Considerando campi ulteriori a quelli previsti dal *Common Log file Format* si potrebbe raffinare la definizione delle sessioni analizzando tali valori. Conoscendo browser e sistema operativo usati, ad esempio, si potrebbero considerare appartenenti a sessioni diverse richieste effettuate utilizzando software differenti. Occorre tener conto, però, che i client non sempre inviano dati veritieri al server, bisognerebbe quindi valutare l'incidenza di tali casi.

L'utilizzo del campo *referrer*, previsto dall'*Extended Log File Format,* sarebbe un notevole apporto per analisi relative alle sessioni, in quanto questo campo indica la risorsa dalla quale è stata effettuata la richiesta. Semplificherebbe, quindi, il riconoscimento dei percorsi seguiti dagli utenti e la distinzione di sessioni diverse: una sessione sarebbe creata cercando richieste successive il cui URL di richiesta coincida con l'URL del referrer della richiesta successiva.

5.5. Percorsi

Fino ad ora sono state definite particolari istruzioni che consentono di scegliere due pagine (A e B) e determinare tutte le sessioni in cui è stata visitata in un certo istante la pagina A e in un istante successivo la pagina B. Si può stabilire, inoltre, che le due pagine siano visitate consecutivamente, cioè selezionare le sole sessioni in cui vi è almeno una richiesta della pagina A, seguita dalla richiesta della pagina B (facendo riferimento solo a pagine, cioè trascurando le richieste per gli oggetti contenuti nella pagina A). In questo caso le pagine A e B vengono considerate come un unico nodo e in futuro si potrebbero ripetere analisi di questo tipo considerando nodi costituiti da tre o più nodi. Sarebbe quindi possibile definire un percorso, cioè un insieme di pagine visitate consecutivamente, e stabilire quali percorsi e con quali frequenze vengono seguiti prima e dopo la visita del percorso definito inizialmente, ottenendo risultati del tipo:

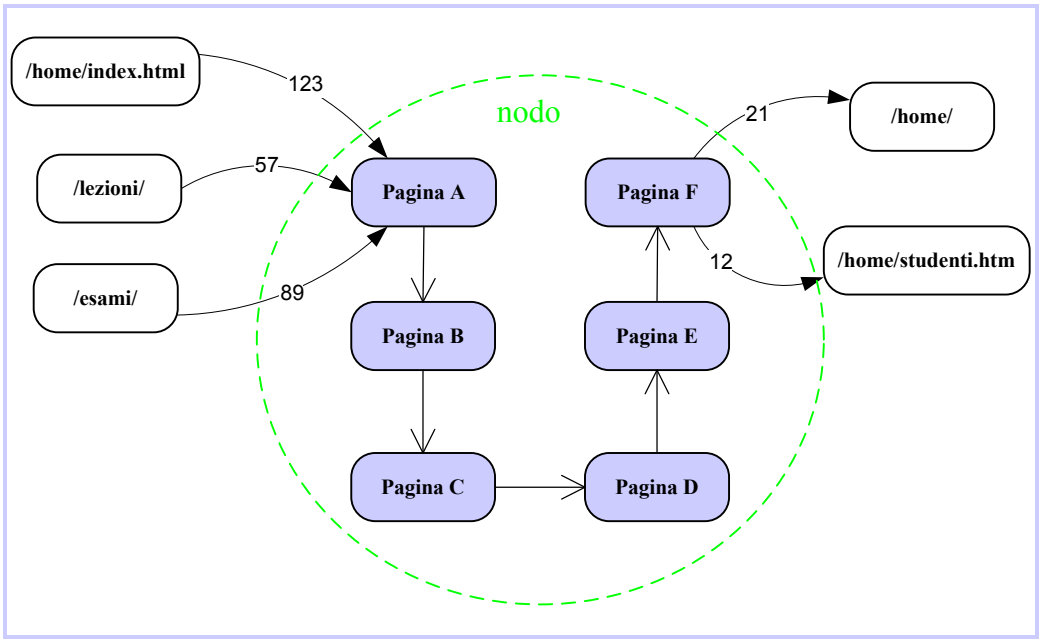


Figura 16 – Percorso passante per sei pagine

5.6. Profili utenti

Le richieste registrate dal server web non sempre vengono inviate da persone che in quel momento stanno utilizzando il client, ma in alcuni casi sono inviate da programmi. *Spider*, *crawler* o *web bot* sono programmi che automaticamente effettuano, in base a determinati criteri, una serie di richieste di file ad un server web, allo scopo di indicizzare i contenuti di quel sito per conto di un motore di ricerca. Le richieste provenienti da *spider* possono incidere fortemente sulla rilevazione del traffico generato da un sito. Per tale motivo sarebbe opportuno che gli accessi prodotti da *spider* venissero evidenziati in modo che se ne possa tener conto e non risultino così falsati i valori relativi alle visite ricevute da parte di utenti umani[27].

Sarebbe interessante anche distinguere le richieste provenienti da proxy. Organizzazioni con un largo numero di utenti (grandi aziende, università, provider, ecc.), infatti, spesso usano un *proxy server* e un *firewall* per proteggere la loro rete interna da intrusioni. Questo significa che le loro reti sono logicamente separate dal resto del mondo, ed è il proxy server che permette la comunicazione tra la loro rete locale e l'esterno. In questi casi tutte le richieste effettuate dai dispositivi della rete locale vengono inoltrate ai server web dal proxy, dunque in tutte le richieste sarà presente un identificativo di quest'ultimo piuttosto che dei singoli dispositivi.

[27] http://www.regione.emilia-romagna.it/sin_info/lineeguida/12_statistiche_glossario_termini.htm

Bibliografia

- Büchner A.G., Anand S.S., Mulvenna M.D., Hughes J.G., *Discovering Internet Marketing Intelligence through Web Log Mining*, 1998.
- He D. e Göker A., *Detecting session boundaries from Web user logs*, 2000.
- Kitsuregawa M., Toyoda M., Pramudiono I., *WEB community mining and WEB log mining: Commodity Cluster based Execution*, 2002.
- Ling C., Gao J., Zhang H., Qian W., Zhang H., *Mining Generalized Query Patterns from Web Logs*, 2001.
- Nanopoulos A., Katsaros D., Manolopoulos Y., *Exploiting Web Log Mining for Web Cache Enhancement*.
- Nanopoulos A., Manolopoulos Y., *Finding Generalized Path Patterns for Web Log Data Mining*, 2000.
- Wall L., Christiansen T. e Schwartz R.L., *Programming Perl*, O'Reilly Associates, seconda edizione, 1996.
- Wang Y., *Web Mining and Knowledge Discovery of Usage Patterns*, 2000.
- http://www.ace.net.nz/tech/TechFileFormat.html, 17 Ottobre 2005.
- http://www.analog.cx, 17 Ottobre 2005.
- http://www.bacuslabs.com/WsvlCLF.html, 17 Ottobre 2005.
- http://bertola.eu.org/icfaq/domini.htm, 17 Ottobre 2005.
- http://www.corsolinux.it/testi/perl/analog/utilizzo_di_analog.jsp, 17 Ottobre 2005.
- http://www.cpan.org/, 17 Ottobre 2005.
- http://www.e-conomy.it/Risorse/e-intelligence/data_mining.htm, 17 Ottobre 2005.
- http://www.graphviz.org/, 17 Ottobre 2005.
- http://www.mach5.com/, 17 Ottobre 2005.
- http://www.mrunix.net/webalizer/, 17 Ottobre 2005.
- http://www.perl.com/, 17 Ottobre 2005.
- http://www.perl.org/, 17 Ottobre 2005.

- http://www.pc-facile.com/glossario/sessione/, 17 Ottobre 2005.
- http://www.regione.emilia-romagna.it/sin_info/lineeguida/12_statistiche_glossario_termini.htm, 17 Ottobre 2005.
- http://www.reportmagic.org/, 17 Ottobre 2005.
- http://www.sawmill.net/, 17 Ottobre 2005.
- http://awstats.sourceforge.net/, 17 Ottobre 2005.
- http://logminer.sourceforge.net/, 17 Ottobre 2005.
- http://www.urchin.com/, 17 Ottobre 2005.
- http://www.webric.it/italiano/analisi_logs.html, 17 Ottobre 2005.
- http://www.webtrends.com/, 17 Ottobre 2005.
- http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html, 17 Ottobre 2005.
- http://www.w3.org/TR/WD-logfile.html, 17 Ottobre 2005.