Badigara: un sistema distribuito per la consultazione dei Bandi di Gara

Franco Accordino *(Consorzio Pisa Ricerche)*

Alfredo Ceccarelli *(CNUCE - Istituto del CNR)*

Rapporto Interno C96-19

Pisa, Agosto 1996

Indice

1. Introduzione

Nell'ambito del progetto "Rete Civica" del Comune di Pisa, realizzato in collaborazione con il Consorzio Pisa Ricerche, si è ritenuto utile e necessario rendere disponibili alle imprese e ditte che agiscono sul territorio locale le informazioni di loro specifico interesse. Il rapporto descrive l'analisi, il progetto e l'implementazione di una Banca Dati (BD) [Albano 85] detta nel seguito "Badigara", ove si trovano memorizzati i dati di sintesi relativi ai bandi di gara (lavori, forniture, servizi) ed il testo completo di ciascun bando.

Il sistema distribuito Badigara utilizza il modello cliente-servente (modello asimmetrico di comunicazione in cui esiste un processo servente in grado di fornire servizi, ed altri processi, clienti, che li richiedono [Stevens 90]) per accedere alla BD. Il sistema è costituito di *personal computer* e stazioni di lavoro, interconnessi tra loro tramite rete locale o geografica [Tanenbaum 91]). Esso può essere quindi pensato come un sistema integrante un processo cliente, un processo servente ed il meccanismo che permette loro di comunicare.

Il servente gestisce le richieste inoltrate dal cliente e interagisce con il gestore della BD, mentre il cliente è utilizzato da un utente per comporre una interrogazione e visualizzarne i risultati.

In questo rapporto viene fatto esplicito riferimento al servente HTTP *Apache* (una versione riveduta e corretta del servente HTTP *NCSA*), mentre come piattaforma di riferimento viene utilizzata una *Sun Sparc Station 5* con sistema operativo *Sun Solaris 2.4* [SunSolaris 94].

Il cliente, software usato per accedere al servente e quindi alla BD, è il *browser Netscape* della *Netscape Corporation*. Esso consente una buona visualizzazione grafica ed è supportato da molteplici Sistemi Operativi (Unix, Windows, MacOs, ecc...).

Per accedere alla BD possono dunque essere usate le seguenti piattaforme: *computer* Apple Macintosh, *personal computer* o stazione di lavoro.

Per la ricerca ed il recupero delle informazioni viene utilizzato il sistema *Fulcrum* della *Fulcrum Technologies Inc.* [Fulcrum 94] in quanto il sistema di indicizzazione sviluppato consente alte prestazioni.

La prima parte del rapporto (Capitolo 2) contiene l'analisi dei requisiti dell'utente.

La parte successiva (Capitolo 3) è rivolta all'utente finale del sistema distribuito Badigara, e descrive le modalità di accesso alla BD tramite la rete di comunicazione *Internet* [Krol 94] e come formulare la ricerca dei documenti.

Nella terza parte (Capitolo 4) sono riportate le caratteristiche principali degli strumenti, di archiviazione e ricerca dei documenti, usati nel progetto.

Nella parte finale (Capitoli 5 e 6) vengono riportate rispettivamente l'infrastruttura di rete del sistema distribuito e lo schema funzionale dello stesso.

2. Analisi dei requisiti.

In ottemperanza alla Legge 241/90 sulla trasparenza negli Enti Pubblici è stato necessario rendere disponibili le informazioni relative ai "Bandi di Gara" (Lavori, Forniture, Servizi) ed ai testi dei medesimi pubblicati all'Albo Pretorio dell'Ente. Con il servizio Rete Civica , qualunque cittadino in posseso di Personal Computer oppure tramite i punti di riferimento prestabiliti (ufficio informazioni, sedi circoscrizionali), può consultare le informazioni.

Attraverso alcuni colloqui con gli esperti del centro elaborazione dati del Comune è stato definito il contesto del problema. Per la gestione automatizzata delle varie tipologie di gara viene utilizzata una procedura sviluppata dal servizio elaborazione dati. La procedura è implementata su un server M480-20 Olivetti con sistema operativo OS/2 Microsoft e software per la rete tipo LAN [Martin 94]. La procedura, sviluppata in ambiente Clipper DB-IV rel.5.1, produce i seguenti archivi:

GADESC:	questa tabella contiene tutti i dati identificativi della pratica.
GAISTR:	contiene tutti i dati relativi all'istruttoria della pratica.
GPTAB1:	tabella relativa ai titpi di gara (licitazione privata, gare ufficiose, appalti concorso).
GPTAB4:	tabella relativa ai servizi proponenti.
GPTAB8:	tabella relativa allo stato di istruttoria della gara.

Una seconda procedura estrae dagli archivi i dati relativi alle seguenti gare: licitazioni private e appalti concorso. Per queste gare sono creati 3 files aventi tutti lo stesso tracciato record relativamente alle gare per l'appalto di Lavori, alle gare per l'appalto di Forniture ed alle gare per l'appalto di Servizi.

Per ogni gara di ciascun tipo di appalto, è stato creato un file di sintesi in cui sono riportate alcune informazioni chiave del relativo testo. I dati di sintesi sono stati convertiti in codice HTML [Lemay 95] secondo le specifiche indicate dal sistema *Fulcrum* per evidenziare le zone di ricerca. In Appendice A viene riportato lo schema di conversione. I dati così ottenuti vengono successivamente caricati nella BD Badigara.

I dati completi di ciascuna gara, invece, vengono convertiti in codice HTML puro e collegati alle rispettive sintesi tramite un *link* ipertestuale.
In questo modo è possibile recuperare il testo completo di una gara attraverso la ricerca dei documenti di sintesi effettuata tramite il motore di ricerca *Fulcrum*. La ricerca per navigazione tramite Netscape consente ugualmente di accedere al testo completo di ciascuna gara.

Le informazioni archiviate possono essere ritrovate tramite l'interfaccia grafica descritta nel capitolo successivo.

3. Modalità di accesso e di ricerca del sistema distribuito Badigara.

3.1 Generalità

Le Banche Dati del Comune di Pisa si propongono di raccogliere e di rendere facilmente accessibili a tutti i cittadini
ed in particolare alle imprese e ditte che operano sul territorio locale le informazioni di loro specifico interesse.

La BD Badigara è quella più completa e fornisce un quadro delle opportunità offerte nel campo delle varie gare di appalto effettuate dalla giunta amministrativa. Essa viene continuamente aggiornata e può essere raggiunta attraverso l'ufficio informazioni e le sedi circoscrizionali collegate in rete al servizio.

Un'altra banca dati, quella contenente le delibere emanate dalla giunta di amministrazione, è attualmente in fase di studio.

3.2 Modalità di accesso al sistema distribuito Badigara

Le interrogazioni alla BD possono essere eseguite in modo semplice e veloce tramite il collegamento alla macchina che ospita i dati.

L'accesso avviene tramite una interfaccia grafica visualizzata nell'ambiente del browser Netscape *(browser* della rete *Internet*). L'interfaccia è realizzata tramite un modulo HTML dove vengono inserite le informazioni oggetto della ricerca e contenente i bottoni per l'esecuzione dei relativi comandi. Questo tipo di interfaccia può essere utilizzata solamente con terminali grafici.

3.3 Struttura di una interrogazione

Il codice che realizza l'interfaccia grafica è riportato in Appendice B. La sua esecuzione è automatica nel momento in cui viene fatto il collegamento al *server* del Comune, *(http://www.comune.pisa.it/search.html),* e comporta la visualizzazione del seguente modulo di ricerca:

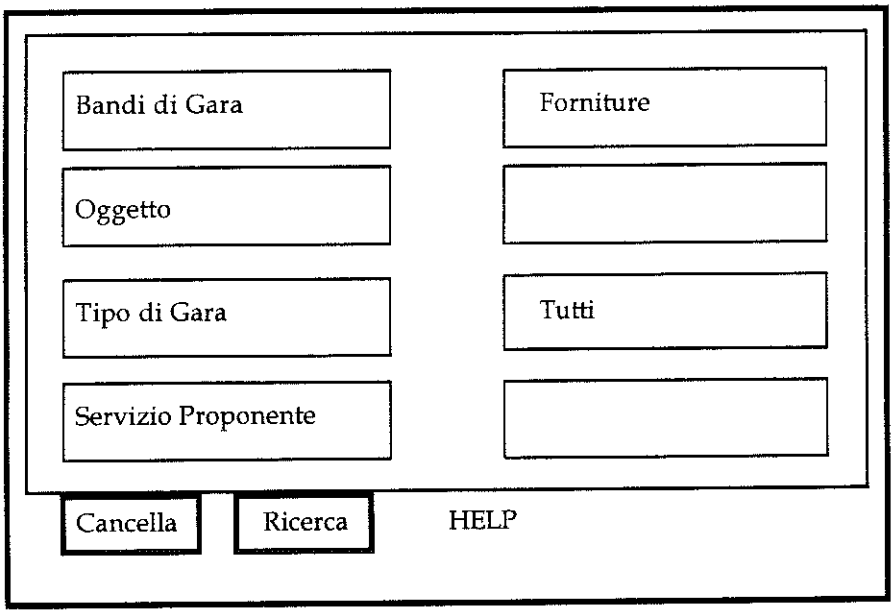

fig. 1 - Modulo di ricerca

I documenti di sintesi archiviati nella BD sono selezionati in base ai valori specificati nelle chiavi di ricerca del modulo, cioè:

- Bandi di Gara
- Oggetto
- Tipo di Gara
- Servizio Proponente

Vediamo dunque quali sono i valori che possono essere assegnati a ciascuna chiave.

Bandi di Gara	specifica il nome della BD dove si desidera effettuare la ricerca.

I documenti sono archiviati in tre distinte BD a seconda della tipologia del Bando di Gara. La BD Forniture si riferisce ai bandi che hanno come oggetto l'acquisizione di apparecchiature varie (acquisto di beni in generale: Personal Computer, ecc...) la BD Lavori contiene i documenti relativi ai Bandi di Gara avente come oggetto i Lavori (costruzione o ristrutturazione di edifici, strade, ecc...), infine la BD Servizi si riferisce ai bandi di gara avente come oggetto i Servizi offerti dall'amministrazione comunale (prestazione d'opera in genere: mensa, pulizia edifici, assistenza agli anziani, ecc...).

Il sistema imposta automaticamente la chiave al valore Forniture, dunque la ricerca dei documenti sarà effettuata nella BD Forniture. Nel caso in cui si voglia effettuare la ricerca in una delle altre disponibili, è necessario selezionare la finestra indicata. Viene visualizzata la lista delle BD che consente di specificare quella desiderata.

Oggetto	consente di specificare una o più parole come criterio di selezione dei documenti appartenenti alla BD specificata nella chiave precedente.

La parola codificata deve necessariamente, appartenere all'insieme delle parole che formano l'Oggetto del Bando di Gara. Se l'Oggetto non è noto, il campo può essere lasciato senza valore, in tal caso la chiave non costituisce elemento per la selezione dei documenti.

Tipo di Gara — questa chiave serve ad indicare il tipo di gara del bando.

Il sistema imposta la chiave di ricerca con il valore TUTTI. Nel caso in cui si vogliono selezionare solo i documenti relativi agli Appalti Concorso oppure alla Licitazione Privata, è sufficiente selezionare la finestra e scegliere il Tipo di Gara desiderato.

Servizio Proponente — indica il Servizio che ha proposto il Bando di Gara.

Anche in questo caso il sistema associa alla chiave il valore iniziale TUTTI. Saranno forniti i documenti senza distinzione del Servizio Proponente. Nel caso in cui si vogliono conoscere solamente i documenti relativi ai Bandi di Gara proposti dal CED, dal Servizio Tecnologico e Sicurezza oppure dal Servizio Edilizia Pubblica, si deve selezionare il valore nella lista associata a questa finestra.

A questo punto, la ricerca è stata formulata. Si può premere il bottone Invia per inoltrare la ricerca al sistema oppure annullare la stessa tramite il bottone Cancella. In quest'ultimo caso viene riproposto il modulo per l'inserimento di nuovi valori. Il bottone HELP consente di avere in linea le informazioni per codificare correttamente il modulo di ricerca.

Inviata la ricerca, il sistema visualizza i documenti di sintesi che soddisfano i valori specificati nelle chiavi. Riportiamo nella fig. 3 un esempio del formato di stampa relativo alla interrogazione specificata nella fig. 2.

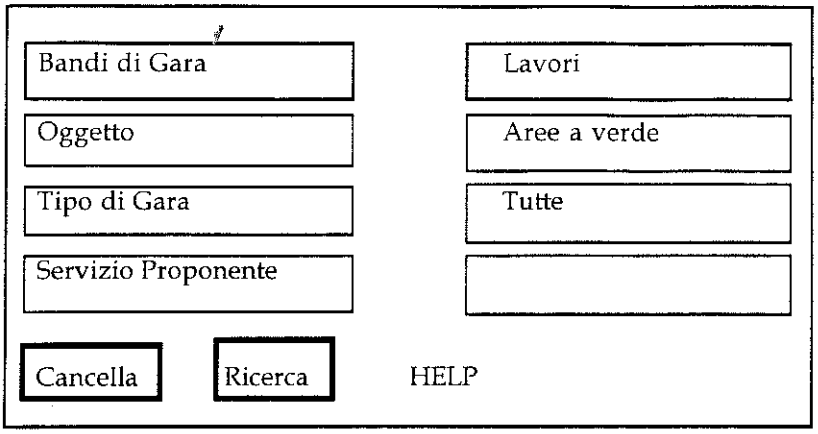

fig. 2 - Esempio di interrogazione della BD

Comune di Pisa **13/06/96** **Risultato della ricerca: 1 elemento trovato**	
OGGETTO:	Lavori di arredo urbano consistenti nella manutenzione straordinaria aree a verde viale delle Piagge, piazza Giusti, e via Don Bosco - sostituzione panchine
ORGANO:	GC
NUMERO DELIBERA:	1543
DATA DELIBERA:	28-09-95
TIPO GARA:	Licitazione Privata
IMPORTO:	41740000
SERVIZIO PROPONENTE:	Urbanizzazione primaria
DATA PUBBLICAZIONE:	27-06-96
DATA SCADENZA:	17-07-96
TESTO DELLA DELIBERA	

fig. 3 - Esempio di stampa di un documento di sintesi

Nella riga di intestazione viene visualizzato il numero di documenti di sintesi che sono stati trovati nella BD e che soddisfano alle condizioni di ricerca specificate. L'ultima riga di ogni documento di sintesi è il link al testo della relativa delibera. Dunque è necessario fare un click sulla stringa per visualizzare il testo completo della delibera stessa.

4. Strumenti di archiviazione e ricerca dei documenti

Per l'archiviazione e la ricerca dei documenti, abbiamo valutato le capacità di alcuni strumenti in grado di fornire l'uso di una semplice interfaccia di utente ed un sistema di *text-retrieval* molto veloce.

Il servente HTTP *Apache* è il sistema utilizzato per la comunicazione in rete Internet con il cliente e attraverso il quale viene raggiunto il gestore della BD.

Il cliente, software usato per accedere al servente e quindi alla BD, è il *browser Netscape* della Netscape Corporation. Esso consente una buona visualizzazione grafica ed è supportato da molteplici Sistemi Operativi (Unix, Windows, MacOs).

Per la ricerca ed il recupero delle informazioni viene utilizzato il sistema *Fulcrum* della *Fulcrum Technologies Inc.* in relazione alla sua efficacia nella gestione di grosse quantità di informazioni.

4.1 Il sistema WWW

Il progetto WWW (*World Wide Web,* detto anche *W3*) è nato nei laboratori del CERN di Ginevra agli inizi degli anni '90 allo scopo di consentire l'accesso ed il recupero di informazioni allocate fisicamente in varie parti del mondo, in maniera *efficiente* e *user-oriented.*

Fino ad allora l'accesso alla "grande rete mondiale" era un privilegio di pochi utenti, dotati di una preparazione adeguata alla comprensione e all'uso dei protocolli esistenti (FTP, Gopher, ecc...).

Con l'avvento del WWW, è oggi possibile fare in modo che anche gli utenti meno esperti possano comunicare interattivamente dalla scrivania del proprio ufficio o abitazione.

L'interfaccia user-oriented che è stata concepita per il WWW viene implementata da un programma *cliente*, detto *browser*, che consente di visualizzare documenti composti da testo, immagini, animazioni e altri oggetti multimediali.

In un documento alcuni degli oggetti (detti *link* o *anchor*) possono essere evidenziati per indicare che, attraverso una semplice selezione con il mouse, è possibile richiamare altri documenti situati su qualche altro computer (possibilente remoto), detto *servente*. Gli oggetti ritrovati sono normalmente altri documenti ipermediali, così la "navigazione" può continuare per un certo numero di passi fino al ritrovamento delle informazioni desiderate.
E' anche possibile, e spesso conveniente, utilizzare un'altra modalità di accesso ai documenti offerta dal WWW: l'accesso attraverso l'interrogazione per parole chiave.
In questo caso l'utente invia la richiesta di esecuzione di una ricerca digitando su una appropriata interfaccia una condizione di ricerca. Il servente risponde inviando tutti i documenti che soddisfano quella condizione.

E' anche possibile fare in modo che dati già esistenti possano essere "pubblicati in rete" senza ulteriori sforzi. Per esempio, in uno dei casi più semplici, il servente può generare una "vista" ipertestuale che rappresenta la struttura di una *directory* esistente.

Infine, dal punto di vista dell'utente vale la pena sottolineare che attraverso il modello ipermediale adottato dal WWW è possibile costruire interfacce utente espressive, con un elevato grado di personalizzazione, e nello stesso tempo sufficientemente uniformi e

semplici da non richiedere nessun particolare *training* da parte degli utilizzatori.

Dal punto di vista più tecnico gli elementi fondamenteli del sistema WWW sono:

- Un sistema di indirizzamento che permetta di riferire univocamente e uniformemente gli oggetti nello spazio Web.

- Un protocollo di rete usato dai clienti e dai serventi WWW che consenta caratteristiche e prestazioni altrimenti non raggiungibili.

- Un linguaggio ipertestuale, interpretabile dai clienti WWW, usato per la descrizione di testo, immagini, animazioni, e tutto ciò che può costituire un'interfaccia espressiva e facile da usare.

- Una tecnica standard per la realizzazione di complesse interazioni client-server consentendo cosi' l'integrazione del sistema WWW con pacchetti applicativi esistenti quali i DBMS, gli IRS, ecc....

Descriviamo più in dettaglio questi elementi.

4.1.1 Il sistema di indirizzamento URL

Gli *Universal Resource Locator* (URL) sono stringhe di lunghezza variabile utilizzate come indirizzi di entità nel Web. Essi sono "universali" nel senso che contengono tutte le informazioni richieste per identificare gli elementi nell'insieme *globale* delle risorse di rete. Con l'introduzione degli URL è quindi possibile, attraverso un comune schema di indirizzamento, accedere ad oggetti normalmente acceduti usando di volta in volta protocolli differenti (NNTP, FTP, WAIS, ecc...).
Un URL contiene informazioni su:

- Il protocollo da usare per accedere la risorsa (http, ftp, wais, telnet, ...),
- Il nome simbolico del servente nel dominio Internet (oppure il suo indirizzo IP)
- L'eventuale nome della porta attraverso la quale viene erogato il servizio,
- Il *path* della risorsa all'interno del servente,
- Gli eventuali dati da passare alla risorsa (quali per esempio i parametri di una ricerca), se la risorsa è un programma da eseguire sul servente.

Gli URL possono inoltre essere rappresentati in forma *assoluta* o *relativa*. Un URL assoluto indica univocamente la posizione di un oggetto nello spazio globale di indirizzamento della rete; un URL relativo si riferisce invece alla posizione di un oggetto rispetto ad un altro URL. Gli URL relativi consentono di sviluppare la struttura a grafo degli iperlink in modo indipendente, favorendo così la portabilit^ degli oggetti da un servente ad un altro, senza per questo dover modificare i collegamenti reciproci tra gli oggetti stessi.

Vale la pena ricordare che gli URL sono un aspetto centrale del sistema WWW. Il fatto che sia così facile indirizzare un oggetto ovunque esso si trovi è essenziale per la scalabilità del sistema e per l'indipendenza delle informazioni dalla topologia della rete.

4.1.2 Il protocollo di comunicazione HTTP

HyperText Transfer Protocol (HTTP) è un protocollo per la trasmissione di dati progettato appositamente per rendere semplice ed efficiente lo scambio di dati ipermediali. Infatti, quando l'utente naviga per la rete, gli oggetti visitati vengono ritrovati in rapida successione anche se allocati su serventi posti a distanze fisicamente ragguardevoli.

Il protocollo HTTP, come la maggior parte dei protocolli adottati sulla rete Internet, si basa sul modello *cliente-servente*. Esso è simile (nella sua forma orientata al trasporto di testo) ai più comuni protocolli di livello applicativo Internet quali FTP (File Transfer Protocol), NNTP (Net News Transfer Protocol), ecc..

Tuttavia, a differenza degli altri protocolli, esso è *stateless* ovvero è eseguito su una connessione TCP [Comer 90] che viene mantenuta soltanto per la durata di una operazione.
Il modello stateless risulta più appropriato poiché un link da un oggetto prelevato da un server S1 può condurre ad un altro oggetto che può essere situato su un servente S2 in generale diverso da S1.

HTTP è usato per accedere informazioni memorizzate in un illimitato ed estendibile insieme di formati (binario, carattere, ecc...). Per raggiungere questo scopo, il cliente invia la lista dei formati che può riuscire ad interpretare, mentre il servente dal canto suo invia i dati codificati in uno dei formati che può produrre. In altre parole, attraverso una fase di mediazione trasparente all'utente, le controparti possono scambiarsi dati in formati proprietari di qualunque natura, senza la necessità di effettuare conversioni in formati standardizzati.

Una transazione HTTP è composta da 4 stadi:

1. apertura della connessione,
2. il cliente invia una richiesta (*request*),
3. il servente risponde al cliente (*response*),
4. chiusura della connessione.

Il risultato di ogni transazione è lo scambio di un insieme di dati (tipicamente un file), oppure l'elaborazione sulla macchina servente dei dati passati dal cliente.

Il cliente contatta il servente inviando un *HTTP Request Header*, eventualmente seguito dai dati che il cliente vuole mandare.
In risposta il servente invia un *HTTP Response Header*, seguito dagli eventuali dati di risposta.
Nel caso più comune l'*HTTP Request Header* si riferisce alle richiesta di un file, situato in una qualche directory nel server, contenente dati in un determinato formato.

Per esempio se:

```
<A
HREF="http://www.comune.pisa.it/gare/lavori/gara1.html">TEST
O DELLA GARA</A>
```

è un link ad un documento, allora quando l'utente seleziona con il mouse il testo evidenziato "TESTO DELLA GARA", il cliente invia al servente le seguenti informazioni:

```
GET /gare/lavori/gara1.html HTTP/1.0
Accept: text/plain
Accept: text/html
.....
Accept: */*
User-Agent: Netscape 1.11
```

Una *HTTP Request Header* lanciata da un cliente inizia sempre con un codice di operazione conosciuto come *metodo*, seguito dal path che localizza univocamente la risorsa dentro il servente e dalla versione del protocollo usato.

I metodi più comunemente usati sono GET, POST e HEAD.
Il metodo GET viene in genere utilizzato dai browser per la semplice richiesta di documenti oppure, quando i dati da passare al servente sono limitati, nella richiesta di esecuzione di programmi. Il metodo GET generalmente preserva lo stato della rete, ovvero non modifica il contenuto dei documenti presenti nello spazio Web.
Il metodo POST viene usato per l'*attachement* di dati alla richiesta di un oggetto, ed è particolarmente utile per spedire ai serventi grosse quantità di dati immessi dall'utente attraverso una *form*.

Infine, il metodo HEAD viene impiegato per ritrovare informazioni nello header della risorsa richiesta, tipicamente la data di ultima modifica, il tipo della risorsa stessa, ecc...
Gli altri campi dell'*HTTP Request Header* informano il servente dei differenti tipi di dato che il cliente è in grado di accettare.

Essi sono espressi nella forma di *header Content-Type*. L'insieme degli *header* ammessi è un'estensione delle Miltipurpose Internet Mail Extentions (MIME).
Altri header importanti sono: *User-Agent*, che comunica al servente il nome e la versione del programma che effettua la richiesta, *Referer* che fornisce al servente la URL del documento da cui è stata effettuata la richiesta, ecc...
L'*HTTP Request Header* deve essere sempre concluso con una linea contenente uno e un solo CRLF (Carriage Return + Line Feed).

Il servente, ricevuto l'*HTTP Request Header*, risponde inviando le informazioni sullo stato della transazione, il tipo dei dati spediti, alcune informazioni opzionali, e i dati veri e propri.

Per esempio, la risposta del servente alla richiesta sopra esemplificata potrebbe essere:

HTTP/1.0 200 OK
Date: Monday, 15-Jul-96 09:12:44 MET
Server: HTTPD Apache/1.3
MIME-version: 1.0
Content-Type: text/html
Last-modified: Friday, 12-Jul-96 15:41:38 MET
Content-length: 2823

<HTML>
<HEAD>
<TITLE>GARA n. 3</TITLE>
</HEAD>
<BODY>
..... [corpo del documento] .....
</BODY>
</HTML>

Il cliente in base al *tipo* dei dati ricevuti effettua le opportune azioni.
Se per esempio riceve un file contenente codice HTML (il caso più comune), allora si limita ad interpretare e visualizzare il contenuto del file. Un altro caso abbastanza diffuso è la ricezione di un file in formato compresso; in questo caso il cliente apre una finestra di dialogo dove l'utente può specificare la directory dove salvare il file (*downloading*).
E' importante sottolineare che il cliente determina il tipo dei dati esclusivamente dallo *header Content-Type* ricevuto.

4.1.3 Il linguaggio di programmazione HTML

Nonostante le possibilità offerte da HTTP di negoziare il formato dei dati scambiati, il sistema WWW è stato dotato di un linguaggio di base per lo scambio di informazioni ipertestuali, detto HTML (*HyperText Markup Language*).

HTML è stato progettato per essere semplice e potente, in modo da poter essere facilmente usato da chiunque desideri pubblicare informazioni sul Web. Esso permette di realizzare documenti composti da testo, liste strutturate, immagini di vari formati, tabelle, iperlink, ecc....

Il linguaggio supporta un insieme di comandi (elementi o *tag*) che consentono di accettare dati di input dall'utente di inviarli al servente. Tra questi, uno dei più importanti è il tag <FORM>, che permette di realizzare sofisticati modelli per l'immissione di dati (*form*). Una *form* è composta da uno o più campi di input, ognuno dei quali è caratterizzato da un *nome* e da un *tipo* (text, textarea, select, radio-button, check-button, hidden, ecc...). Il *valore* del campo viene fornito dall'utente attraverso la digitazione di testo oppure attraverso la selezione con il mouse di zero o più opzioni. Infine, in ogni *form* è generalmente presente un pulsante che consente di avviare la trasmissione dei dati al servente.

I dati vengono spediti al servente come un insieme di elementi della forma *nome=valore*, opportunamente codificati e concatenati (con il carattere '&' come separatore) in un unica stringa detta *query string*. In particolare, vengono codificati tutti i caratteri non ASCII, gli spazi e la maggior parte dei simboli speciali. Questo aspetto è importante perché è compito del programma invocato (vedi par. 4.1.4) decodificare e separare i dati ricevuti.

L'URL al quale sottoporre i dati viene specificata nell'elemento <FORM>. é responsabilità esclusiva del cliente codificare i dati inseriti e spedirli al servente.

In Appendice B viene riportato il codice che visualizza la *form* per effettuare la query nella banca dati Badigara.

L'uso dell'HTML non è vincolato al protocollo HTTP. Infatti, esso può essere applicato nella E-Mail ipertestuale, nelle news e in qualsiasi altro settore in cui risulti efficiente ed espressivo adottare uno schema ipertestuale.
Recentemente il linguaggio ha subito una crescente e continua evoluzione che ha portato alla introduzione di nuovi elementi capaci di realizzare interfacce dotate di effetti e funzionalità sempre più sofisticate. Uno di questi elementi è il tag <APPLET> che consente di integrare nell'ipertesto piccoli programmi (*applet*) scritti nel linguaggio JAVA.

4.1.4 L'interfaccia di comunicazione CGI

La *Common Gateway Interface* (CGI) specifica il modo in cui i dati di input, forniti dall'utente e recapitati al servente attraverso il protocollo HTTP, vengono passati al programma che li elabora. Tale programma, comunemente denominato *programma CGI* o *script CGI*, deve risiedere sulla macchina dove è in esecuzione il servente HTTP.
La fig. 4 riassume il flusso dei dati tra processi in una generica transazione HTTP.

- Il cliente HTTP invia al servente la richiesta di un documento HTML oppure la richiesta di esecuzione di un programma accodando gli eventuali dati immessi dall'utente (arco 1);

- Se la richiesta si riferisce ad un semplice documento HTML il servente accede immediatamente al documento e lo offre al cliente (arco 4);

- Altrimenti, se la richiesta si riferisce all'esecuzione di un programma sul servente, inoltra la richiesta al corrispondente script CGI (arco 2);

- Lo script CGI elabora i dati ricevuti e restituisce la risposta al servente che li riflette al cliente attraverso il protocollo HTTP (archi 3 e 4);

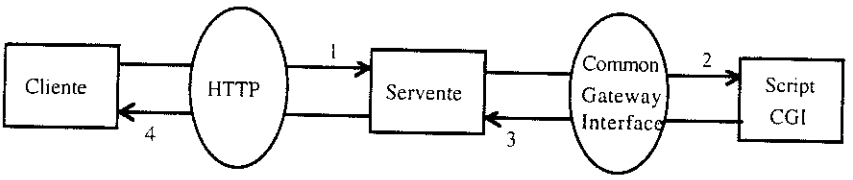

fig. 4

I dati ricevuti dal sevente possono essere passati al programma CGI attraverso variabili di ambiente (questo è il caso dei dati inviati con il metodo GET), oppure attraverso lo standard input (usato quando il cliente invia i dati al servente con il metodo POST).

[Metodo GET vs Metodo POST]

Il servente in ogni caso utilizza variabili di ambiente per comunicare al programma CGI informazioni circa le caratteristiche del cliente e del servente stesso.

Il programma CGI restituisce al servente i risultati scrivendo sullo standard output le appropriate direttive seguite dai dati da restituire al cliente, oppure inviando direttamente al cliente un *HTTP response header* seguito dai dati veri e propri. In questo rapporto verrà utilizzato soltanto il secondo metodo. Sarà sempre cura del programma CGI inviare un *HTTP response header* al cliente.

La *Common Gateway Interface* definisce un insieme di variabili di ambiente che vengono passate dal servente allo script CGI al momento della chiamata. Riportiamo di seguito alcune tra le più importanti variabili con il relativo significato.

- SERVER_SOFTWARE: Il nome e la versione del servente HTTP che ha invocato il CGI script.

- SERVER_NAME: Il nome dell'host (o il numero IP) come appare nell'URL che indirizza lo script CGI.

- SERVER_PROTOCOL: La versione del protocollo HTTP al quale fa riferimento il servente.

- REMOTE_HOST: Il nome dell'host che ha effettuato la richiesta.

- REQUEST_METHOD: Il metodo usato per effettuare la richiesta (GET, POST, ecc...).

- CONTENT_TYPE: Il tipo dei dati inseriti dall'utente e accodati dal cliente alla richiesta (solo metodo POST).

- CONTENT_LENGHT: La dimensione dei dati inviati dal cliente.

- HTTP_ACCEPT: I tipi MIME che il cliente è in grado di interpretare.

- HTTP_USER_AGENT: Nome e versione del cliente.

- QUERY_STRING: I dati che seguono il simbolo '?' nell'URL che ha riferito lo script.

Lo schema funzionale di un generico script CGI è un pipeline composto da quattro fasi alcune delle quali possono anche mancare. La fig. 5 riporta un esempio di script CGI.

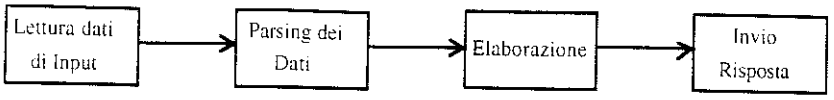

fig. 5

Nella prima fase il programma legge dall'ambiente (se il metodo è il GET) oppure dallo standard input (se il metodo è il POST) l'input pervenuto dal cliente attraverso il protocollo HTTP.

Una volta letto, l'input viene analizzato e tradotto in strutture dati locali al CGI (seconda fase). Come accennato nel paragrafo 2.1.3, i dati giungono al servente come una lista di coppie *Nome=Valore* separate dal carattere '&'. Il parsing dei dati di input consiste nel separare le varie coppie e memorizzare in corrispondenti variabili locali i valori associati ai campi di input. I valori immessi dall'utente diventano cosi' disponibili in altrettante variabili nel programma CGI per essere elaborate.

L'elaborazione (terza fase) prende in input le variabili di utente, le variabili di ambiente e tutto ciò che può servire alla realizzazione del servizio e produce un output che viente spedito al cliente sotto forma di codice HTML per poter essere visualizzato (quarta fase). La risposta deve in ogni caso avvertire l'utente del risultato dell'elaborazione, eventuali errori inclusi.

È importante osservare che quando un programma CGI termina l'ambiente di valutazione viene distrutto. Questo rende impossibile far comunicare esplicitamente due o più istanze diverse di uno script CGI (conservazione dello stato). Il problema puo' essere risolto facendo comunicare le due istanze in due differenti modi:

a) attraverso i campi di tipo *hidden* inseriti nelle form HTML di risposta.

b) attraverso il file system del servente;

La soluzione *a* _ possibile solo se le due istanze sono lanciate da richieste provenienti dallo stesso cliente; inoltre è poco efficiente perché implica una ritrasmissione dei dati tra una chiamata e l'altra.
La soluzione *b* è più generale ed efficiente in presenza di grosse quantità di dati da preservare, ma può condurre (a differenza della soluzione *a*) a programmi CGI poco concisi e strutturati.

L'interfaccia CGI rappresenta un metodo standard per implementare funzionalità altrimenti ottenibili soltanto modificando l'architettura e il codice del servente HTTP. Essa consente lo sviluppo di applicazioni caratterizzate da un elevato grado di interattività in maniera efficiente, semplice e modulare.

4.2 Il sistema Fulcrum.

Nei tradizionali DBMS (*Data Base Management System*), i dati sono prevalentemente costituiti da un elevato numero di campi di vario tipo e di dimensione limitata. Essi sono organizzati in una struttura gerarchica, reticolare o relazionale in un sistema di proprietà chiamato *database*, ed acceduti tramite una chiave (*keyword*) che identifica univocamente ogni elemento (o record).

Diversamente, nei sistemi IRS (*Information Retrieval System*) i dati sono costituiti da insiemi di documenti (o frammenti di testo) di vario formato memorizzati in file di sistema, sistemi di immagazzinamento di dati, ecc... L'informazione testuale è fondamentalmente non strutturata e logicamente costituita da insiemi di parole (*word*) di varia lunghezza.

Nel sistema Badigara, per archiviare i documenti di sintesi avremmo potuto utilizzare un DBMS; infatti i documenti di sintesi sono caratterizzati da una limitata quantità di dati di piccole dimensioni. Tuttavia, la presenza di grosse quantità di dati come il testo completo di ciascun bando di gara, ci ha indotto a scegliere un

IRS quale sistema per l'archiviazione e la ricerca dei documenti.

In particolare la nostra scelta è ricaduta sul sistema Fulcrum, un pacchetto software che attraverso un indice di parole, presenti nei documenti, lo rende particolarmente efficace nella gestione di grosse quantità di informazioni.

Le principali caratteristiche del sistema Fulcrum sono:

- un motore di ricerca ad alte prestazioni e multipiattaforma per l'indicizzazione e la ricerca su grandi quantità di documenti memorizzati in vari formati. Il motore viene chiamato SearchServer.
- il motore di indicizzazione e ricerca incorpora un'architettura *client/server* che consente all'applicazione che gira sulla macchina dello user (*client*) accedere al SearchServer che gira su un host remoto (*server*). Con questa architettura, le funzioni di interfaccia utente sono generalmente eseguite sui nodi *client* mentre le funzioni di indicizzazione e recupero sono eseguite sui nodi *server*. Solo il testo e altri dati di interesse dello user sono trasferiti *on demand* al sistema *client*.
- il linguaggio di interrogazione è basato su una estensione verso il *text-retrieval* di un sottoinsieme dell'SQL (*Structured Query Language*) standard chiamato SearchSQL.

  permette una integrazione trasparente con i sistemi informativi esistenti. I dati memorizzati nei documenti non necessitano, in generale, di una rigorosa strutturazione. Esso accede ai documenti nel loro formato originale attraverso i *Text Readers* di documento. Alcuni vantaggi di quest'ultimo approccio sono:

  - non è necessario convertire o immagazzinare i documenti in un separato database sebbene il SearchServer possa cercare i documenti anche in un database esterno

- non è necessario conoscere il formato dei documenti che gli utenti stanno cercando
- gli utenti possono simultaneamente ricercare per documenti creati in differenti formati

- l'architettura SearchServer supporta i *Text Readers* di database per integrare i dati che sono già memorizzati in un DBMS. L'applicazione può usare SearchServer per accedere direttamente a questa informazione.
- fornisce l'accesso alle funzionalità del SearchServer attraverso le *SearchServer API (Application Program Interface)*
- l'architettura del SearchServer include un back-end API per text-readers che consente ai programmatori in linguaggio C [Kernighan 88] scrivere e integrare un text reader dello user con SearchServer.
- utilizza il modello dei sistemi di gestione dei database relazionali, per creare le strutture delle tabelle che rappresentano le collezioni dei documenti. Ogni riga della tabella rappresenta un riferimento logico ad un documento.

Il SearchServer consente di avere tutti i vantaggi e la potenza dei DBMS relazionali esistenti con in più le capacità di ricerca di testo. Questa architettura è caratterizzata dai seguenti aspetti:

- fornisce un più veloce e efficace accesso alle informazioni testuali in un database
- consente di accedere ad esistenti oggetti tipo testo senza doverli ricopiare
- integra dati testuali e numerici

4.2.1 L'architettura di Fulcrum

Il SearchServer presenta una interfaccia uniforme per la ricerca ed il recupero dei dati sia che sono memorizzati localmente o in remoto. Esso fornisce l'interfaccia API *(Application Program Interface)* attraverso la quale gli statements SearchSQL sono eseguiti ed i risultati sono ritornati all'applicazione.

L'architettura del sistema consiste di componenti architetturali e componenti funzionali.

Le componenti architetturali, riportate in fig. 6 includono:

- SearchServer API
- SearchSQL
- SearchServer driver

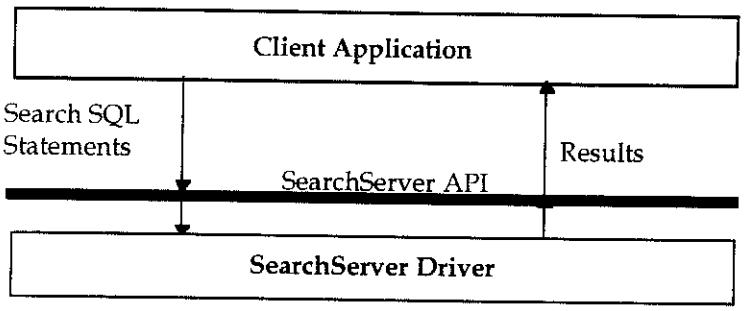

fig. 6 - Architettura del SearchServer

Il SearchServer supporta le *open systems* API che costituiscono l'interfaccia primaria al SearchServer.

Le funzioni API sono organizzate nei seguenti tre gruppi funzionali:

- funzioni di sessione e connessione
- funzioni per la gestione degli *statements*
- funzioni per il recupero dei dati

La ricerca, l'indicizzazione e le funzionalità per la gestione delle tabelle sono accedute attraverso SearchSQL. SearchSQL è un sottoinsieme esteso di ISO e ANSI SQL è il linguaggio di interfaccia standard per accedere ai sistemi di database (vedi par. 4.2.4) Il motore di ricerca fornisce un miglioramento di alcune prorietà per aumentare la flessibilità e la potenza delle applicazioni *text-retrieval.*

Il SearchServer supporta anche ambienti dove un ODBC Driver Manager potrebbe essere presente (come i sistemi Unix e OS/2). Lo scopo del driver manager ODBC è provvedere a:

- caricare dinamicamente un driver su richiesta dell'applicazione
- fornire più drivers per gestire differenti *data sources* intermediare tra l'applicazione del cliente e il driver.

Le componenti funzionali del SearchServer riportate nella fig. 7 ,sono:

- *Search Server driver*
- *Local SearchServer*
- *Remote SearchServer*
- *Text Readers*

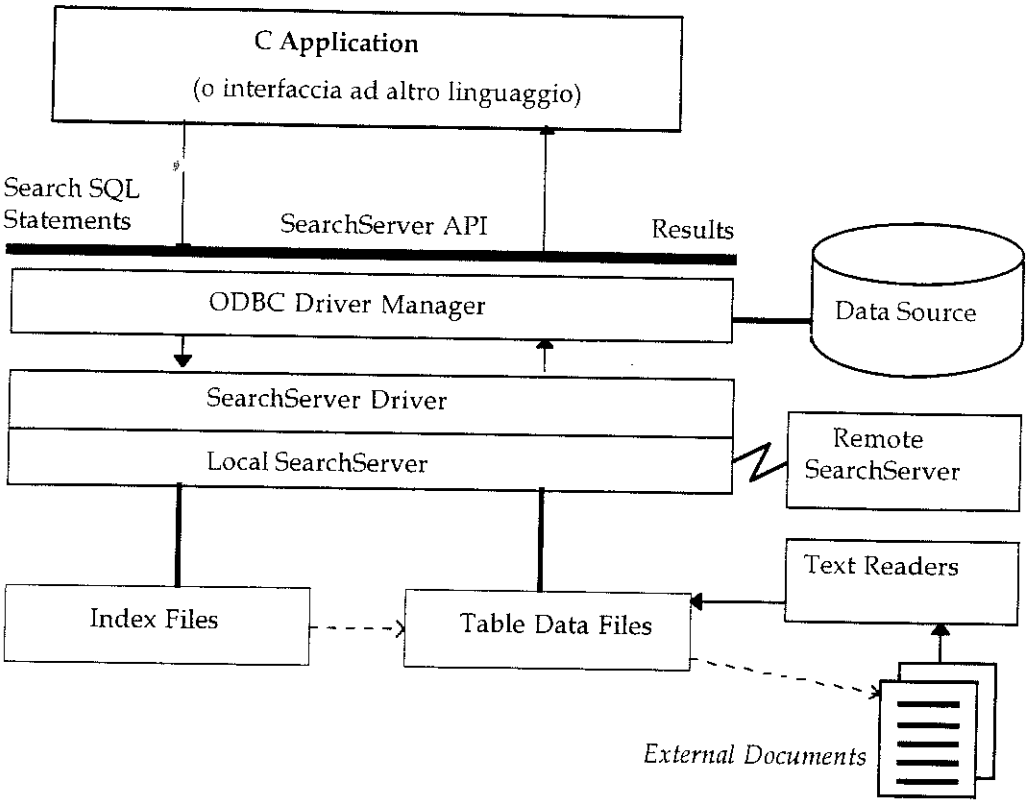

fig. 7 - Architettura del SearchServer e componenti funzionali

Il *SearchServer Driver* contiene il gestore della connessione, il gestore degli statements SearchSQL ed il sistema di recupero dei dati. Il gestore della connessione è responsabile della gestione delle sessioni, delle connessioni e della gestione degli errori. Il gestore degli statements SearchSQL è responsabile della gestione per l'esecuzione ed i risultati degli statements SearchSQL. Infine il sistema per il recupero dei dati è responsabile per ottenere i dati e ritornare questi all'applicazione.

Le componenti del *SearchServer Driver* sono mostrate nella fig. 8.

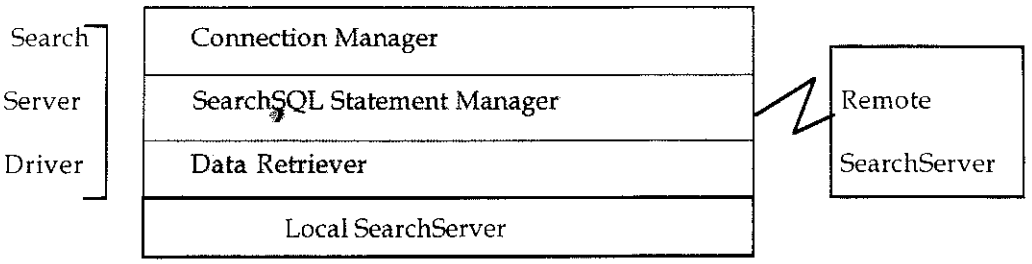

fig.8 - Componenti del SearchServer driver

Nell'insieme, le componenti del *driver SearchServer* gestiscono il *link* al SearchServer remoto e la chiamata al SearchServer locale per accederi i dati locali.

Il *SearchServer* locale fornisce gli strumenti di accesso ai dati locali ed implementa il modello della tabella.

Le componenti funzionali per accedere alle tabelle locali sono:

- *table manager*
- *data reader*
- *indexing engine*
- *search engine*

Il gestore della tabella è usato per creare e mantenere le tabelle, mentre il lettore dei dati estrae i dati dalle tabelle e dai documenti per indicizzare e recuperare i dati. Il motore di indicizzazione mantiene gli indici ai dati, mentre il motore di ricerca legge gli indici per selezionare i dati che soddisfano il criterio di ricerca. Questi componenti sono mostrati nella fig. 9.

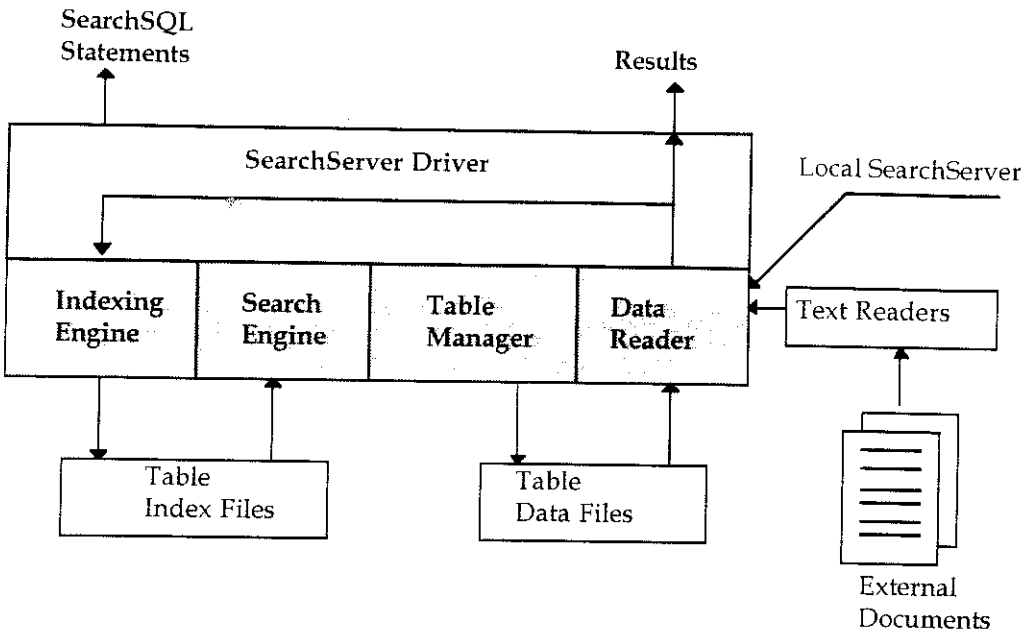

fig. 9 - Componenti del SearchServer Locale

Il SearchServer può gestire le informazioni memorizzate nelle sue tabelle, e può accedere alle informazioni memorizzate in files di sistemi operativi esterni e in storage system più evoluti come i database relazionali. Gestisce direttamente i testi strutturati e i dati memorizzati nelle sue tabelle.

Una architettura text reader generalizzata è usata per accedere oggetti tipo *external text*. Esso usa i text readers di documenti per leggere nel loro formato originale e locazione esistente, e text readers di database per leggere text memorizzati in database.

Il componente *Table data files* contiene i dati che sono direttamente gestiti dal SearchServer, mentre *l'Index files* contiene gli indici per una tabella. Quest'ultimo consente la ricerca rapida del contenuto della tabella e del testo esterno relativo.

Un documento esterno è un file testuale che ha il suo proprio formato nativo. Esso può essere creato da un word processor o text editor, o essere il prodotto di altre applicazioni, come e-mail, news o WWW.

Il SearchServer remoto gestisce i dati remoti per conto dell'applicazione SearchServer. Esso esegue le stesse funzioni (e contiene le stesse componenti) del SearchServer locale.

I text readers interpretano i testi in un particolare formato altrimenti indecifrabile dal SearchServer. Per esempio, un documento esterno scritto in Microsoft Word può essere traslato *on demand* tramite un text reader di documento nel formato riconosciuto dal SearchServer.

Un text reader di documento è invocato dal SearchServer ogniqualvolta un documento esterno è indicizzato o recuperato. Un text reader di database è usato per leggere i testi memorizzati in un database system DBMS.

Fulcrum fornisce alcuni text readers standard per il riconoscimento e trattamento dei più comuni standard. Nel caso in cui fosse necessario costruire un proprio text reader, il SearchServer fornisce un back-end text reader API per questo scopo.

4.2.2 Il modello logico dei dati

Il modello logico dei dati è una astrazione degli oggetti che l'applicazione richiede. Esso consente una organizzazione dei dati in strutture ad alto livello come: schemi, tabelle, righe, colonne, zone e domini. I dati sono organizzati logicamente in modo da massimizzare la potenza e la flessibilità delle operazioni text-retrieval disponibili all'applicazione.

Il SearchServer presenta un modello di dati che è familiare al programmatore SQL. Infatti adotta il concetto matematico di relazione n-m, equivalente al concetto tradizionale di tabella in cui i dati sono organizzati in righe e colonne della tabella.

Nel sistema Fulcrum sono definite tre classi di colonne:

Reserved columns - sono colonne riservate, create dal sistema e utilizzate dal SearchServer per implementare il proprio modello di dati, e dall'utente per poterlo usare. Il SearchServer utilizza queste colonne e i loro valori per fornire informazioni di controllo ai motori di ricerca e di indicizzazione e per fornire informazioni all'utente. Queste colonne sono sempre presenti in una qualsiasi tabella del SearchServer.

Internal columns - è la classe di colonne definibili dall'utente.

External text column - questa colonna contiene il puntatore al testo esterno che compone il corpo di un documento. Il testo esterno non è memorizzato dal SearchServer, ma si trova nel suo formato nativo in un file del sistema operativo. Per accedere a questi documenti dinamicamente e in modo trasparente, il SearchServer fornisce un meccanismo chiamato "text reader". I documenti possono essere in vari formati, compreso il formato proprietario, i più comuni formati word process, o nel semplice formato ASCII. Il SearchServer fornisce sia un insieme standard di text readers che dei tool per svilupparne di nuovi a seconda delle esigenze applicative. I text reader operano in cascata, utilizzando un meccanismo identico ad una pipeline (vedi fig. 10).
Il testo prodotto è uno stream di caratteri che può essere riconosciuto dal SearchServer.

Il formato che riconosce il SearchServer è chiamato FTDF (*Fulcrum Technologies Document Format*).

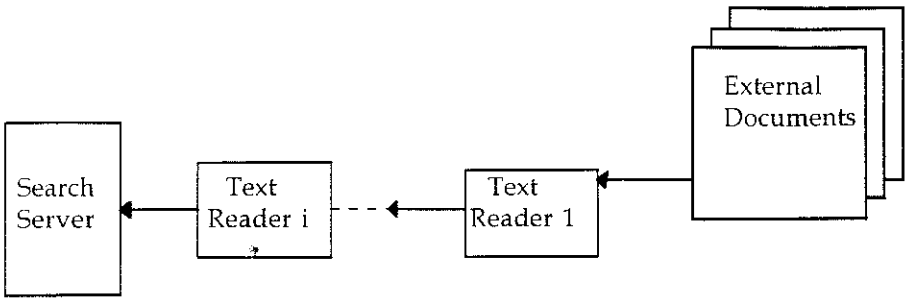

fig. 10 - Stream dei Text Readers

Il modello logico dei dati fa ancora uso di concetti e termini familiari ai programmatori SQL. Nel sistema SearchServer troviamo i seguenti nuovi concetti:

- schema
- zona
- dominio

Lo *schema*, o dizionario dei dati, è la descrizione logica dei dati che si vogliono includere nella tabella. Lo statement CREATE SCHEMA permette di definire la tabella, le sue colonne, gli attributi delle colonne, le zone e i domini. In Appendice C sono riportati gli statements SearchSQL utilizzati per creare lo schema e le tabelle della BD Badigara.

Le *zone* consentono di distinguere regioni del testo (come il nome di un capitolo o una intestazione) all'interno di una colonna di testo più ampia. Una zona è un parte di una colonna di testo. Le zone danno perciò maggiore flessibilità alla tabella consentendo una maggiore strutturazione dei dati. I dati nella zona diventano una sezione indipendente usata solo per scopi di ricerca. Questo migliora la precisione delle query e riduce il tempo di ricerca.

Dunque, è possibile dividere il testo di un documento in differente zone per distinguere tra una parola che appare in un titolo (per esempio), e la stessa parola che appare nel corpo del testo.

Solo una colonna contenente testo può essere segmentata. Il concetto di zona non può essere applicato a colonne tipo *date* e *numeric*.

Una colonna può contenere una o più zone. Se non viene specificata alcuna zona, viene assegnata dal sistema una zona di default il cui numero è lo stesso del field number della colonna. Qesta è considerata una semplice colonna per distinguerla dalla colonna segmentata in cui sono definite le zone.

Una zona è definita con la clausola CREATE ZONE dello statement CREATE SCHEMA. Ogni zona ha un unico nome e numero di zona associato. Questi numeri si riferiscono alle zone segnate (sequenze di controllo che delimitano le istanze delle zone) nel testo.

Il *dominio* è un tipo di dato definito dall'utente che è costruito su un tipo di dato predefinito. Esso è usato per associare zone con una colonna, per cambiare *l'index mode* per un tipo di dato predefinito o per cambiare il *data lenght* di default. Quando un dominio è usato per associare una lista di zone con una colonna, una ricerca sulla colonna è effettivamente una ricerca su tutte le zone della colonna. I domini sono usati per raggruppare le zone in una colonna.

Le zone sono assegnate alle colonne usando la clausola CREATE DOMAIN dello statement CREATE SCHEMA. Le zone sono raggruppate insieme nella definizione del dominio e referenziate nella definizione della colonna della clausola CREATE TABLE. Quando un dominio viene assegnato a una colonna, tutte le zone del dominio sono assegnate a quella colonna. Successivamente, in fase di ricerca, un match in alcune zone della colonna equivale a un match nella colonna stessa.

4.2.3 Il modello fisico dei dati

I dati sono fisicamente organizzati per massimizzare l'efficienza di ricerca e recupero. Il modello fisico dei dati consiste nei file ODBC, nei file per la gestione della tabella, nei file di supporto alla tabella e infine nei documenti esterni.

I file ODBC contengono i data source, le definizioni dei drivers che sono usati dal gestore dei driver OBBC e i drivers per stabilire una connessione a un insieme di dati. I file di gestione della tabella contengono i dati, gli indici della tabella e i riferimenti ai documenti esterni. I file di supporto alla tabella migliorano le operazioni del SearchServer.

Quando viene creata una tabella, il SearchServer genera i file di gestione della tabella in una locazione specificata nella definizione del data source oppure nel parametro dello statements CREATE SCHEMA. I file sono inizialmente vuoti fino a quando la tabella non contiene i dati. I dati nella tabella sono caricati in base alla definizione dello schema. Nel processo di indicizzazione, il SearchServer indicizza i dati memorizzati nei file di gestione della tabella e i documenti esterni ad essa associati. L'indicizzazione consiste nel registrare nei file indice della tabella, le occorrenze di ogni parola di testo, numero e data

Al momento della ricerca, il SearchServer pone il risultato della ricerca in una tabella di lavoro. Le informazoni sono poste nella tabella di lavoro a fronte dell'operazione di query per le informazioni medesime. La tabella di lavoro contiene i dati che sono direttamente rilevati dalla query. Le tabelle di lavoro sono fisicamente rappresentate come reference in un file temporaneo che esiste per la durata di una sessione di utente.

4.2.4 Il linguaggio SearchSQL

Il linguaggio SearchSQL è un sottoinsieme esteso del linguaggio ANSI SQL, il linguaggio di interfaccia standard per accedere ai database relazionali. Esso fornisce anche un miglioramento di alcune prorietà per aumentare la flessibilità e la potenza delle applicazioni *text-retrieval.*

Gli statements SearchSQL possono essere divisi in tre categorie:

- *Data Definition Language (DDL)* cioè gli statement usati per definire e amministrare una tabella o uno schema, cancellare una tabella esistente, proteggere la tabella da cancellazioni o alterazioni accidentali ed aggiornare l'indice per una tabella. (CREATE, DROP, PROTECT, VALIDATE INDEX).
- *Data Manipulation Language (DML)*, ovvero gli statements per inserire nuove righe in una tabella, aggiornare i valori di colonne selezionate in una o più righe, cancellare una o più righe dalla tabella (INSERT INTO, UPDATE, DELETE FROM).
- *Search and Retrieval Language (SRL)* per effettuare la ricerca in una o più tabelle, specificare le righe e le colonne che debbono essere visualizzate, l'ordine di sort del risultato delle righe, nonchè impostare le opzioni per gli statements che debbono essere eseguiti per la durata di una connessione a uno specifico *data source* (SELECT, SET).

Dunque Search SQL supporta la seguenti funzionalità:

creazione e amministrazione di tabella e schema, operazioni di inserimento, aggiornamento e cancellazioni di dati, operazioni di indicizzazione e di ricerca.

5. Infrastruttura di rete

Lo schema in fig. 11 mostra la infrastruttura di rete tra il Comune di Pisa e il CPR. La rete LAN del Comune di Pisa è composta da un certo numero di host e da un router CISCO 2511 con funzioni di terminal server dotato di linee telefoniche commutate. Attualmente sono attive soltanto 8 delle 16 linee disponibili, utilizzate per la connessione in via sperimentale di alcune circoscrizioni comunali e altri utenti. Uno degli host, una *Sun Sparcstation 5* con sistema operativo *Sun Solaris 2.4* ospita le pagine WWW ed alcuni servizi TCP/IP (E-Mail, FTP, ecc...). La LAN è collegata con linea dedicata CDN a 64 Kbit/sec con il CPR, attraverso il quale avviene l'accesso alla rete geografica Internet.

Presso il CPR è situato l'host (una *Sun Sparcstation 10* con sistema operativo *Sun OS 4.1.3* [SunOS 95] sul quale è installato il servente Fulcrum e dove viene gestita la banca dati Badigara.

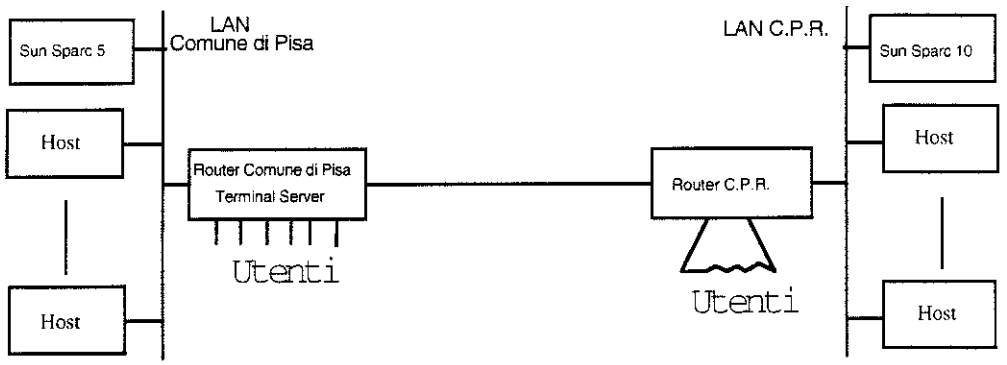

Fig. 11 - Schema della infrastruttura di rete

6. Schema funzionale del sistema distribuito BADIGARA

La fig.12 descrive i principali moduli software che compongono il sistema Badigara. Le zone tratteggiate indicano ognuna un differente host.

Il cliente HTTP (tipicamente un browser HTML come Netscape o Mosaic) può accedere alle gare direttamente attraverso il servente HTTP del Comune di Pisa.

Infatti, nella directory pubblica del servente *www.comune.pisa.it*, troviamo le tre sottodirectory:

- *public_html_path/gare/forniture/sintesi/* (s1)
- *public_html_ path/gare/lavori/sintesi/* (s2)
- *public_html_ path/gare/servizi/sintesi/* (s3)

dove sono contenute le sintesi delle gare, e le tre corrispondenti sottodirectory:

- *public_html_ path/gare/forniture/testo/* (t1)
- *public_html_ path/gare/lavori/testo/* (t2)
- *public_html_ path/gare/servizi/testo/* (t3)

dove sono raccolti i testi completi.

Grazie al meccanismo di generazione automatica del contenuto delle directory (funzionalità che caratterizza le ultime generazioni di serventi HTTP, detta *fancy directory indexing*), è possibile accedere alle sintesi ed al testo delle gare attraverso una "vista ipertestuale" del contenuto delle directory sopra citate (v. par. 4.1.4). Per ogni elemento (sintesi o testo) della risultante "vista ipertestuale" vengono riportati il titolo (come dal campo <TITLE> del codice HTML) e un link al documento vero e proprio.

È importante osservare che mentre i testi completi delle gare (directory t1-t3) vengono tradotti manualmente con il tag HTML <PRE>, la procedura Clipper per l'estrazione delle sintesi (ved. cap. 2.) crea documenti HTML contenente i codici di controllo Fulcrum per la delimitazione delle zone di ricerca. Per rendere direttamente accessibile attraverso il WWW il contenuto delle directory s1-s3 è dunque necessaria una fase di pre-elaborazione (*filtering*) che, dato un qualsiasi documento *sintesi-gara.html'* decorato con i codici di controllo Fulcrum, produce un documento *sintesi-gara.html* in formato HTML puro.

Il tipo di accesso fin qui descritto viene detto *accesso per navigazione* (archi 1-4 in fig. 12), ed è praticabile soltanto in presenza di una quantità limitata di documenti. L'accesso per navigazione attraverso le *fancy directory indexing* rappresenta un rudimentale sistema di *fault tollerance* utilizzabile quando il servente Fulcrum è per qualche motivo inaccessibile.

Se il numero di documenti nelle directory diventa elevato, tipicamante dell'ordine delle decine di elementi, allora l'accesso per navigazione può diventare inadeguato. In questi casi viene chiamato in causa il motore di ricerca del sistema Badigara. L'utente compila il modulo di ricerca (vedi par. 3.3) prelevandolo ancora dal servente del Comune di Pisa e riceve come risultato la lista di sintesi che soddisfa la condizione di ricerca specificata.

L'URL riportata nel campo <FORM> del modulo HTML fa riferimento allo script CGI "*search*" il cui codice è riportato in Appendice D, installato sul servente HTTP *dbases.cpr.it* (archi 5 e 12).

Lo script CGI si comporta da interfaccia (*Gateway*) tra il WWW e il sistema Fulcrum:

- prende in input i dati inseriti dall'utente,
- costruisce la corrispondente query SSQL,
- sottopone al servente Fulcrum la query costruita,
- ricava i risultati e li presenta all'utente in formato HTML.

Sulla macchina *dbases.cpr.it* sono installati il Fulcrum SearchServer e un servente HTTP. Quest'ultimo è impiegato solo come intermediario (attraverso lo script CGI) tra il cliente WWW e il Fulcrum Search Server (in altre parole sulla macchina *dbases.cpr.it* non vi sono documenti HTML direttamente accessibili),

L'inserimento dei dati avviene in due fasi effettuate *off-line* utilizzando l'interfaccia *execsql* disponibile con il pacchetto Fulcrum:

1. Ogni documento di sintesi prodotto dalla procedura di estrazione viene trasferito sulla macchina *dbases.cpr.it* e collocato (in base al tipo di gara al quale fa riferimento) in una delle seguenti directory :

- *badigara_docs_path/gare/forniture/* (b1)
- *badigara_docs_ path/gare/lavori/* (b2)
- *badigara_docs_ path/gare/servizi/* (b3)

Alle suddette directory vengono associate rispettivamente le tabelle FULCRUM **Gare_F**, **Gare_L** e **Gare_S**, la cui struttura, peraltro uniforme, è riportata in Appendice C. La creazione delle tabelle viene effettuata all'inizio attraverso la shell di Fulcrum *execsql*.

2. Per ogni documento *sintesi-gara.html'* trasferito viene eseguita la query SQL:

INSERT INTO Gare_X (Nome_File , FT_FLIST) VALUES ('sintesi-gara.html', 't:s') ;

dove Gare_X è la directory in cui il documento *sintesi-gara.html'* è stato collocato.

Essendo le tabelle state create con modalità di indicizzazione IMMEDIATE, dopo l'esecuzione della query il documento è immediatamente ricercabile e quindi accessibile via WWW.

L'eliminazione o l'aggiornamento dei documenti nella banca dati avviene sempre in modalità *off-line* eseguendo le corrispondenti query SQL DELETE e UPDATE.

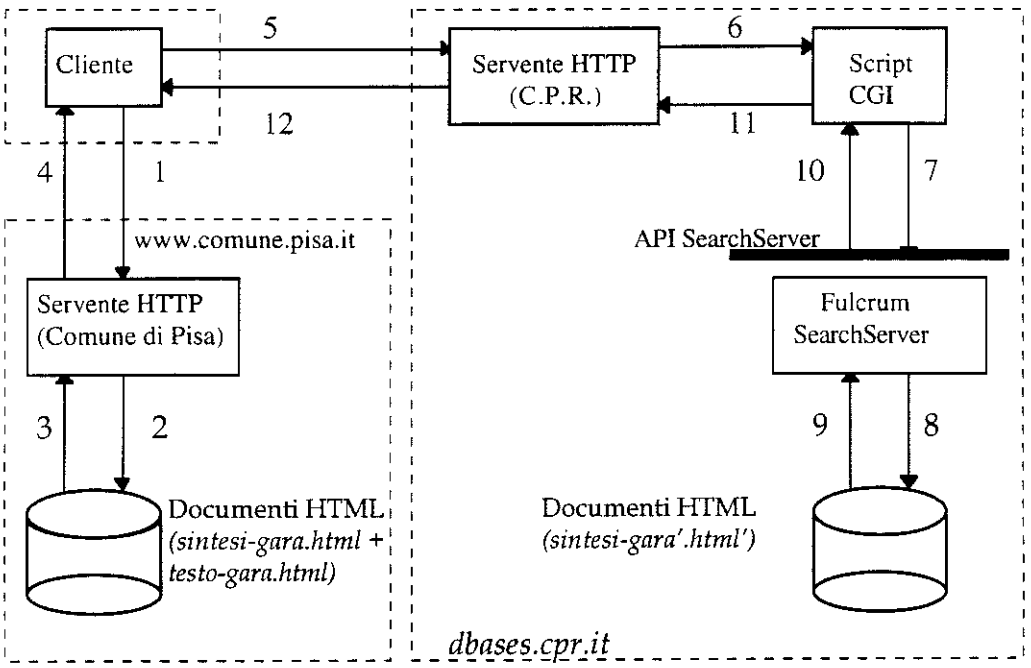

fig. 12 - Architettura software del sistema distribuito Badigara

Fin'ora abbiamo visto la struttura ad alto livello dei processi e le loro interazioni. La fig. 13 descrive la struttura del programma CGI che implementa l'interazione tra il WWW e il sistema Fulcrum.

I dati inseriti dall'utente nella form HTML di ricerca giungono al servente attraverso il protocollo HTTP; Essendo GET il metodo specificato nella form, il servente passa i dati al programma CGI attraverso la variabile QUERY_STRING (v. par. 4.1.4).

Successivamente, viene effettuato il *parsing* della QUERY_STRING.
A tal proposito viene usata una libreria di funzioni C creata *ad hoc* per la manipolazione ad alto livello dei dati di input codificati nella QUERY_STRING (v. par 4.1.1). In particolare per ogni tipo di campo previsto nelle *form* HTML esiste una appropriata funzione che lo converte in una struttura dati locale al programma CGI riportato in Appendice D. La *form* di ricerca *search.html* utilizza soltanto campi di tipo <INPUT>, <SELECT> e <SELECT MULTIPLE>. I campi di tipo <INPUT> vengono convertiti in stringhe C. I campi di tipo SELECT vengono tradotti in coppie *(Opt, i)*, dove *Opt* è un array di opzioni (una opzione per ogni valore previsto nella form HTML) mentre *i* è un intero che indica l'opzione scelta dall'utente. Infine, i campi di tipo e <SELECT MULTIPLE> vengono trattati come coppie *(Opt, I)*, dove *Opt* ha lo stesso significato di prima mentre *I* è un array di interi tale che *I(j)=1* se l'opzione *j* è selezionata, *I(j)=0* altrimenti.

Una volta separati i campi in altrettante strutture dati locali viene effettuato un semplice controllo sintattico. In questa fase vengono anche inserite sequenze di escape nei campi di tipo <INPUT> al fine di prevenire eventuali errori sintattici nella fase di valutazione della query SQL.

A tal proposito occorre osservare che il carattere "'" rappresenta nel SSQL l'inizio e la fine di un pattern di ricerca nelle condizioni espresse con i costrutti LIKE, CONTAINS, ecc....Se il carattere "'" appare in un campo di tipo <INPUT> (il carattere "'" è piuttosto frequente nella lingua italiana, usato come apostrofo o accento) allora perchè venga interpretato correttamente deve essere preceduto un altro carattere "'" (sequenza di escape).

Se durante il parsing della QUERY_STRING è rilevato un errore di sintassi allora viene segnalato all'utente il tipo di errore commesso invitandolo a leggere la pagina di aiuto per la compilazione della form. Se invece non vengono rilevati errori viene costruita una query nel linguaggio SearchSQL.

Occorre osservare che i campi della *form* HTML nel loro insieme vengono considerati in AND logico, le opzioni dei campi <SELECT MULTIPLE> vengono considerate in OR logico, infine, le condizioni espresse dall'utente con le parole chiave AND e OR nei campi di tipo <INPUT> vengono tradotte utilizzando i corrispondenti operatori logici SearchSQL.

Dopo la costruzione della query il programma CGI effettua una connessione con il Fulcrum SearchServer. Ricordiamo che, in generale, per l'interazione con il Fulcrum Search Server viene utilizzata la libreria API di programmi C fornita da Fulcrum. Le funzioni dell'API di Fulcrum che sono state usate nello script CGI possono essere divise in 3 classi: funzioni per la connessione e la disconnessione, funzioni per la valutazione di query (search) e funzioni per il retrieval dei dati.

Se la connessione dà esito negativo (determinato per esempio da un crash del Fulcrum SearchServer oppure da una insufficienza di memoria) allora viene inviata all'utente una risposta appropriata pregandolo di avvertire, via E-Mail, l'amministratore del Fulcrum SearchServer.

Se invece la connessione va a buon fine, la query costruita in precedenza viene inviata al FULCRUM Search Server utilizzando le funzioni dell'API per la valutazione di query.

Il risultato della valutazione della query viene automaticamente immagazzinato in strutture dati locali al CGI. In particolare viene allocato un array avente tante colonne quanti sono i campi specificati nella clausola SELECT e tante righe quanti sono i documenti di sintesi ricercati dal Fulcrum Search Server, soddisfacenti la condizione di ricerca impostata nella clausola WHERE.
Infine, utilizzando le funzioni dell'API di Fulcrum per il recupero dei dati, i documenti di sintesi memorizzati nella struttura dati temporanea vengono acceduti e inviati al cliente.

L'utente vede così una lista di sintesi (*headlines*) attraverso la quale può accedere al testo completo delle gare.

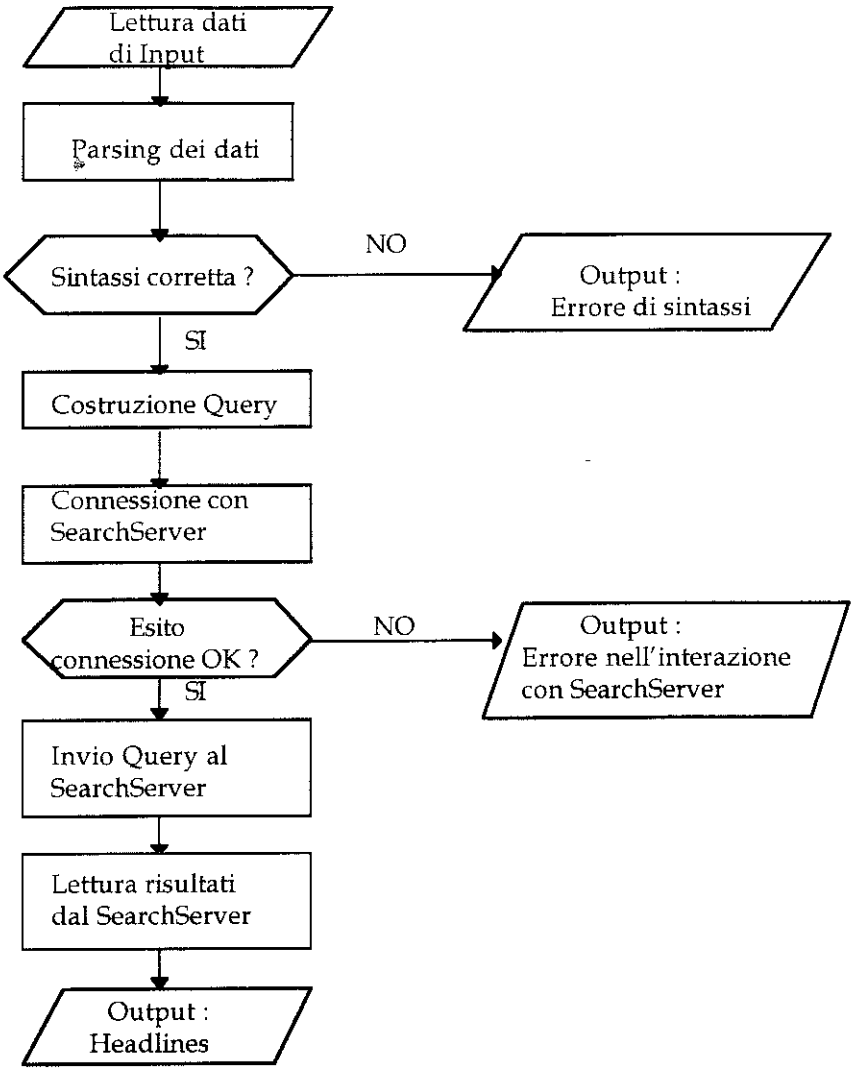

fig. 13 - struttura del CGI che implementa l'interazione tra il WWW ed il sistema FULCRUM.

7. Conclusioni

In questo rapporto tecnico è stato descritto un prototipo di sistema distribuito per la consultazione dei bandi di gara del Comune di Pisa, denominato Badigara. Il sistema è stato inizialmente analizzato nelle sue componenti a livello di requisiti dell'utente, di infrastruttura di rete e di supporto hardware; successivamente ne è stata disegnata l'architettura software e implementate le funzionalità richieste. Nella implementazione del sistema sono state effettuate precise scelte hardware/software suggerite da una preliminare indagine sui principali prodotti (di pubblico dominio e non) presenti sul mercato. Il progetto Badigara è parte integrante del progetto "Rete Civica" del Comune di Pisa, tecnicamente supportato dal Consorzio Pisa Ricerche (CPR).

I punti rimasti aperti sono tanti, anche alla luce delle frequenti evoluzioni cui è sottoposta la rete Internet (e il progetto WWW in particolare) sia dal punto di vista delle nuove tecnologie per la trasmissione dei dati che dal punto di vista degli strumenti software. Tra i più importanti ricordiamo:

- Un sistema di gestione dei dati trasparente agli addetti del CED del Comune di Pisa e del CPR. Infatti, attualmente i dati delle delibere vengono inseriti, modificati e cancellati manualmente. Una soluzione potrebbe essere la implementazione di un sistema client-server che mantenga aggiornati i dati sull'host del Comune di Pisa e sull'host del CPR che ospita la banca dati.

- La installazione del servente che gestisce la banca dati direttamente presso il Comune di Pisa. Questa evoluzione del progetto consentirebbe infatti una semplificazione della infrastruttura sia a livello di rete che a livello software.

- Un adeguato periodo di testing dal punto di vista dell'affidabilità e dell'uso. Quest'ultimo aspetto potrebbe essere valutato attraverso questionari da sottoporre ad un insieme prescelto degli utenti del sistema.

- La creazione di un meta-sistema personalizzabile per la gestione dei bandi di gara di pubbliche istituzioni, eventualmente estendibile anche ad altri dati (delibere, normative, leggi, ecc...).

Appendice A

Formato dei dati di Sintesi

```
<HTML><HEAD><TITLE>OGGETTO</TITLE></HEAD>
<BODY>
<B>OGGETTO:</B>\E[201sOGGETTO\E[s<BR>
<B>ORGANO:</B>\E[202sORGANO\E[s<BR>
<B>NUMERO DELIBERA:</B>\E[203sNUMERO DELIBERA\E[s<BR>
<B>DATA DELIBERA:</B>\E[204sDATA DELIBERA\E[s<BR>
<B>TIPO GARA:</B>\E[205sTIPO GARA\E[s<BR>
<B>IMPORTO:</B>\E[206s\E[8vIMPORTO\E[7v\E[s<BR>
<B>SERVIZIO PROPONENTE:</B>\E[207sSERVIZIO PROPONENTE\E[s<BR>
<B>DATA PUBBLICAZIONE:</B>\E[208sDATA PUBBLICAZIONE\E[s<BR>
<B>DATA SCADENZA:</B>\E[209s\E[8vDATA SCADENZA\E[7v\E[s<BR>

<A HREF="http://www.comune.pisa.it/gare/NOME DIRECTORY/testo/NOME
FILE">TESTO DEL DOCUMENTO</A>
<HR>
</BODY>
</HTML>
```

Esempio di Sintesi di Gara

```
<HTML><HEAD>
<TITLE>LAVORI DI ARREDO URBANO CONSISTENTI NELLA MANUTENZIONE
STRAORDINARIA AREE A VERDE VIALE DELLE PIAGGE, PIAZZA GIUSTI E VIA
DON BOSCO - SOSTITUZIONE PANCHINE</TITLE></HEAD>
<BODY>
<B>OGGETTO.....: </B>\E[201sLAVORI DI ARREDO URBANO
CONSISTENTI NELLA MANUTENZIONE STRAORDINARIA AREE A VERDE VIALE
DELLE PIAGGE, PIAZZA GIUSTI E VIA DON BOSCO - SOSTITUZIONE
PANCHINE\E[s<BR>
<B>ORGANO.....: </B>\E[202sGC\E[s<BR>
<B>NUMERO DELIBERA.....: </B>\E[203s1543\E[s<BR>
<B>DATA DELIBERA.....: </B>\E[204s28-09-95\E[s<BR>
<B>TIPO GARA.....: </B>\E[205sLICITAZIONE PRIVATA\E[s<BR>
<B>IMPORTO.....: </B>\E[206s\E[8v41740000\E[7v\E[s<BR>
<B>SERVIZIO PROPONENTE: .....:</B>\E[207sURBANIZZAZIONE
PRIMARIA\E[s<BR>
<B>DATA PUBBLICAZIONE.....: </B>\E[208s27-06-96\E[s<BR>
<B>DATA SCADENZA.....: </B>\E[209s\E[8v17-07-96\E[7v\E[s<BR>

<A HREF="http://www.comune.pisa.it/gare/lavori/testo/96_16.html">TESTO DEL
DOCUMENTO</A>
<HR>
</BODY></HTML>
```

Appendice B

Interfaccia Grafica che realizza il modulo di ricerca

```
<HTML><HEAD><TITLE>Banca Dati delle Gare</TITLE></HEAD>
<BODY>
<CENTER>
<TABLE><TR>
<TD> <IMG SRC="/img/i-stenp3.gif" WIDTH=30>
<TD><H1>Bandi di gara</H1>
<TD> <IMG SRC="/img/i-stenp3.gif" WIDTH=30>
</TABLE>
Compilare il seguente modulo per effettuare la ricerca
<HR>
<BR>
</CENTER>

<FORM ACTION="http://dbases.cpr.it/com-bin/search" METHOD=GET>
<TABLE BORDER CELLPADDING=2 CELLSPACING=2 ALIGN=CENTER
VALIGN=CENTER>
<TR>
<TD>
<B>Bandi Gara:</B>
<TD>
<SELECT NAME=O_Bando>
 <OPTION VALUE="GareF">Forniture
 <OPTION VALUE="gareL">Lavori
 <OPTION VALUE="gareS">Servizi
</SELECT>

<TR>
<TD>

<B>Oggetto:</B>
<TD>
<INPUT NAME=C_Oggetto SIZE=30 TYPE=TEXT>

<TR>
<TD>
<B>Tipo Gara:</B>
<TD>
<SELECT NAME=O_TipoGara>
 <OPTION VALUE="0">TUTTI
 <OPTION VALUE="1">APPALTO CONCORSO
 <OPTION VALUE="2">LICITAZIONE PRIVATA
</SELECT>
```

```
<TR><TD>
<B>Servizio Proponente:</B>

<TD>
<SELECT NAME=O_ServProp SIZE=3>
<OPTION VALUE="0">Tutti
<OPTION VALUE="1">Affari del Sindaco e della Giunta
<OPTION VALUE="2">Affari Generali
<OPTION VALUE="3">Affari Sociali
<OPTION VALUE="4">Amministrativo LL.PP.
<OPTION VALUE="5">Amministrativo Uso e Assetto del Territorio
<OPTION VALUE="6">Attivita' di Supporto Farmacie
<OPTION VALUE="7">Attivita' Produttive
<OPTION VALUE="8">Attivita' Sportive e Turismo
<OPTION VALUE="9">Biblioteche
<OPTION VALUE="10">Circolazione e Traffico
<OPTION VALUE="11">Contabilita' Generale
<OPTION VALUE="12">Contabilita' Generali
<OPTION VALUE="13">Cultura
<OPTION VALUE="14">Decentramento
<OPTION VALUE="15">Demografico
<OPTION VALUE="16">Economato
<OPTION VALUE="17">Edilizia Privata e Pubblica
<OPTION VALUE="18">Edilizia Pubblica
<OPTION VALUE="19">Elaborazione Dati
<OPTION VALUE="20">Farmacie
<OPTION VALUE="21">Finanze
<OPTION VALUE="22">Legale
<OPTION VALUE="23">Personale
<OPTION VALUE="24">Pianificazione Urbanistica
<OPTION VALUE="25">Politica Tributaria
<OPTION VALUE="26">Polizia Municipale
<OPTION VALUE="27">Prevenzione e Protezione
<OPTION VALUE="28">Pubblica Istruzione
<OPTION VALUE="29">Rapporti Contrattuali
<OPTION VALUE="30">Struttura Annonaria
<OPTION VALUE="31">Tecnologico e Sicurezza
<OPTION VALUE="32">Tutela Ambiente
<OPTION VALUE="33">Urbanizzazione Primaria

</SELECT>

</TABLE>
<TABLE BORDER=0>
<TR><TD><INPUT VALUE="Cancella" SIZE=30 TYPE=RESET>  
<TD><INPUT VALUE="Ricerca" SIZE=30 TYPE=SUBMIT>   
<TD><B><A HREF="/help.html">HELP</A></B></TD>
</TABLE>
</FORM>

<HR><P ALIGN=LEFT>
Per suggerimenti e informazioni sulla banca dati contattare <A
HREF="mailto:servadm@server.comune.pisa.it">servadm@server.comune.pisa.it</A></P>
</BODY></HTML>
```

Appendice C

Codice SearchSQL di definizione dei dati

```
-- ------------------ FORNITURE ------------------

CREATE SCHEMA GareF

CREATE ZONE Oggetto (201) LITERAL
CREATE ZONE Organo (202) NORMAL
CREATE ZONE NumDelib (203) NONE
CREATE ZONE DataDelib (204) NONE
CREATE ZONE TipoGara (205) NORMAL
CREATE ZONE Importo (206) VALUE
CREATE ZONE ServProp (207) NORMAL
CREATE ZONE DataPub (208) NONE
CREATE ZONE DataScad (209) VALUE

CREATE DOMAIN TipoDoc ( Oggetto, Organo, NumDelib , DataDelib, TipoGara , Importo,
ServProp , DataPub, DataScad ) AS APVARCHAR
CREATE DOMAIN TipoNome NONE AS VARCHAR(260)

CREATE TABLE GareF (
    Doc             TipoDoc     32,
    Nome_File       TipoNome    3 )

IMMEDIATE

BASEPATH '/home/dbases/accord/comune/badigara/forniture' ;


-- ------------------ LAVORI ------------------

CREATE SCHEMA GareL

CREATE ZONE Oggetto (201) LITERAL
CREATE ZONE Organo (202) NORMAL
CREATE ZONE NumDelib (203) NONE
CREATE ZONE DataDelib (204) NONE
CREATE ZONE TipoGara (205) NORMAL
CREATE ZONE Importo (206) VALUE
CREATE ZONE ServProp (207) NORMAL
CREATE ZONE DataPub (208) NONE
CREATE ZONE DataScad (209) VALUE

CREATE DOMAIN TipoDoc ( Oggetto, Organo, NumDelib , DataDelib, TipoGara , Importo,
ServProp , DataPub, DataScad ) AS APVARCHAR
CREATE DOMAIN TipoNome NONE AS VARCHAR(260)
```

```
CREATE TABLE GareL (
    Doc             TipoDoc     32,
    Nome_File       TipoNome    3 )

IMMEDIATE

BASEPATH '/home/dbases/accord/comune/badigara/lavori' ;


-- ------------------ SERVIZI ------------------

CREATE SCHEMA GareS

CREATE ZONE Oggetto (201) LITERAL
CREATE ZONE Organo (202) NORMAL
CREATE ZONE NumDelib (203) NONE
CREATE ZONE DataDelib (204) NONE
CREATE ZONE TipoGara (205) NORMAL
CREATE ZONE Importo (206) VALUE
CREATE ZONE ServProp (207) NORMAL
CREATE ZONE DataPub (208) NONE
CREATE ZONE DataScad (209) VALUE

CREATE DOMAIN TipoDoc ( Oggetto, Organo, NumDelib , DataDelib, TipoGara , Importo,
ServProp , DataPub, DataScad ) AS APVARCHAR
CREATE DOMAIN TipoNome NONE AS VARCHAR(260)


CREATE TABLE GareS(
    Doc             TipoDoc     32,
    Nome_File       TipoNome    3 )

IMMEDIATE

BASEPATH '/home/dbases/accord/comune/badigara/servizi' ;
```

```
int built_cond_servprop(char *Cond, int Option, int *esiste)
{
extern char *option_servprop[];
extern char *servprop[];
char SP[100];

if (strcmp(option_servprop[Option], "0") == 0) {
        *esiste=0;
        return(1);
}

put_escapes(servprop[Option], SP);
strcpy(Cond, "ServProp CONTAINS '\0");
strcat(Cond, SP );
strcat(Cond, "'\0" );
*esiste=1;
return(1);

}
```

Libreria di funzioni di utilita'

```
#include "search.h"

char *option[] = {"AND", "OR"};

char *option_importo[] = {"1", "2"};

char *option_tipogara[] = {"0", "1", "2"};

char *option_servprop[] = {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12",
"13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27",
"28", "29", "30", "31", "32", "33"};

char *bandi[] = {"gareF", "gareL", "gareS"};

char *servprop[] = {
"",
"Affari del Sindaco e della Giunta",
"Affari Generali",
"Affari Sociali",
"Amministrativo LL.PP.",
"Amministrativo Uso e Assetto del Territorio",
"Attività' di Supporto Farmacie",
"Attivita' Produttive",
"Attivita' Sportive e Turismo",
"Biblioteche",
"Circolazione e Traffico",
"Contabilita' Generale",
"Contabilità' Generali",
"Cultura",
```

```
"Decentramento",
"Demografico",
"Economato",
"Edilizia Privata e Pubblica",
"Edilizia Pubblica",
"Elaborazione Dati",
"Farmacie",
"Finanze",
"Legale",
"Personale",
"Pianificazione Urbanistica",
"Politica Tributaria",
"Polizia Municipale",
"Prevenzione e Protezione",
"Pubblica Istruzione",
"Rapporti Contrattuali",
"Struttura Annonaria",
"Tecnologico e Sicurezza",
"Tutela Ambiente",
"Urbanizzazione Primaria"
};

int O_Bando;
int O_Oggetto;
int O_TipoGara;
int O_ServProp;

char C_Oggetto[1024];

void reply_error(char *Query)
{
cgiHeaderContentType("text/html");

printf("<HTML><HEAD>");
printf("<TITLE>Risultato della Ricerca </TITLE></HEAD>");
printf("</HEAD><BODY BGCOLOR=\"#FFFFFF\"><CENTER>");
printf("<H3> Risultato della ricerca </H3>");
printf("<H4> Errore nell'interazione con la banca dati </H4>");
printf("<TABLE CELLPADDING=20 CELLSPACING=0>");
printf("<TR><TD><A HREF=\"%s\"> <IMG SRC=\"http://www.comune.pisa.it/img/i-
indi.gif\" BORDER=0></A>", cgiHTTP_Referer);
printf("</TABLE></CENTER></BODY></HTML>");
}

void reply_none(char *Query)
{
cgiHeaderContentType("text/html");

printf("<HTML><HEAD>");
printf("<TITLE>Risultato della Ricerca </TITLE></HEAD>");
printf("</HEAD><BODY BGCOLOR=\"#FFFFFF\"><CENTER>");
printf("<H3> Risultato della ricerca </H3>");
printf("<H4> Nessun elemento trovato </H4>");
printf("<TABLE CEILPADDING=20 CELLSPACING=0>");
```

```
printf("<TR><TD><A HREF=\"%s\"> <IMG SRC=\"http://www.comune.pisa.it/img/i-
indi.gif\" BORDER=0></A>", cgiHTTP_Referer);
printf("</TABLE></CENTER></BODY></HTML>");
}

void put_escapes(char *s, char *t)
{

int i=0;
int j=0;

while (i<strlen(s))
{
if (s[i]=='\'')
 {
  t[j]='\';
 j++;
 }
t[j]=s[i];
i++;
j++;
}
t[j]='\0';
}


void itoa(int n, char *q)
{
int i, sign;
char s[100];
if ((sign=n) <0)
n=-n;
i=0;
do
{
s[i++]=n%10+'0';
} while ((n/=10) > 0);
if (sign<0)
s[i++]='-';
s[i]='\0';
reverse(s);
strcpy(q, s);
}


void reverse(char s[])
{
int c,i,j;
for (i=0, j=strlen(s)-1; i<j; i++, j--)
{
c=s[i];
s[i]=s[j];
s[j]=c;
}
}
```

Bibliografia

[Albano 85]
A. Albano e R. Orsini, "Basi di Dati", *Boringhieri*, Torino, 1985.

[Comer 90]
D. E. Comer, "Internet With TCP/IP", *Prentice-Hall*, 1990.

[Kernighan 88]
B. W. Kernighan, D. M. Ritchie, "Linguaggio C", *Prentice-Hall*, 1987

[Krol 94]
E. Krol, "The Whole Internet Users Guide and Catalog", *Prentice-Hall*, 1994

[Martin 94]
J. Martin, K. K. Chapman, J. Leben, "LAN", *Prentice Halll*, 1994.

[Tanenbaum 91]
A. S. Tanenbaum, "Computer Networks", *Prentice-Hall*, 1991.

[Lemay 95]
Laura Lemay, "Il Manuale HTML", *McGraw-Hill*, 1995.

[Stevens 90]
W. R. Stevens, "UNIX Network Programming", *Prentice Halll*, 1990.

[Fulcrum 94]
Fulcrum SearchServer Introduction to SerachServer, Version 2.0, Fulcrum Technologies Inc., 1994

[SunOS 90]
SunOS Users Manual, Version 4.1.3, Sun Microsystem, 1990

[SunSolaris 94]
SunSolaris Users Manual, Version 2.4, Sun Microsystem, 1994