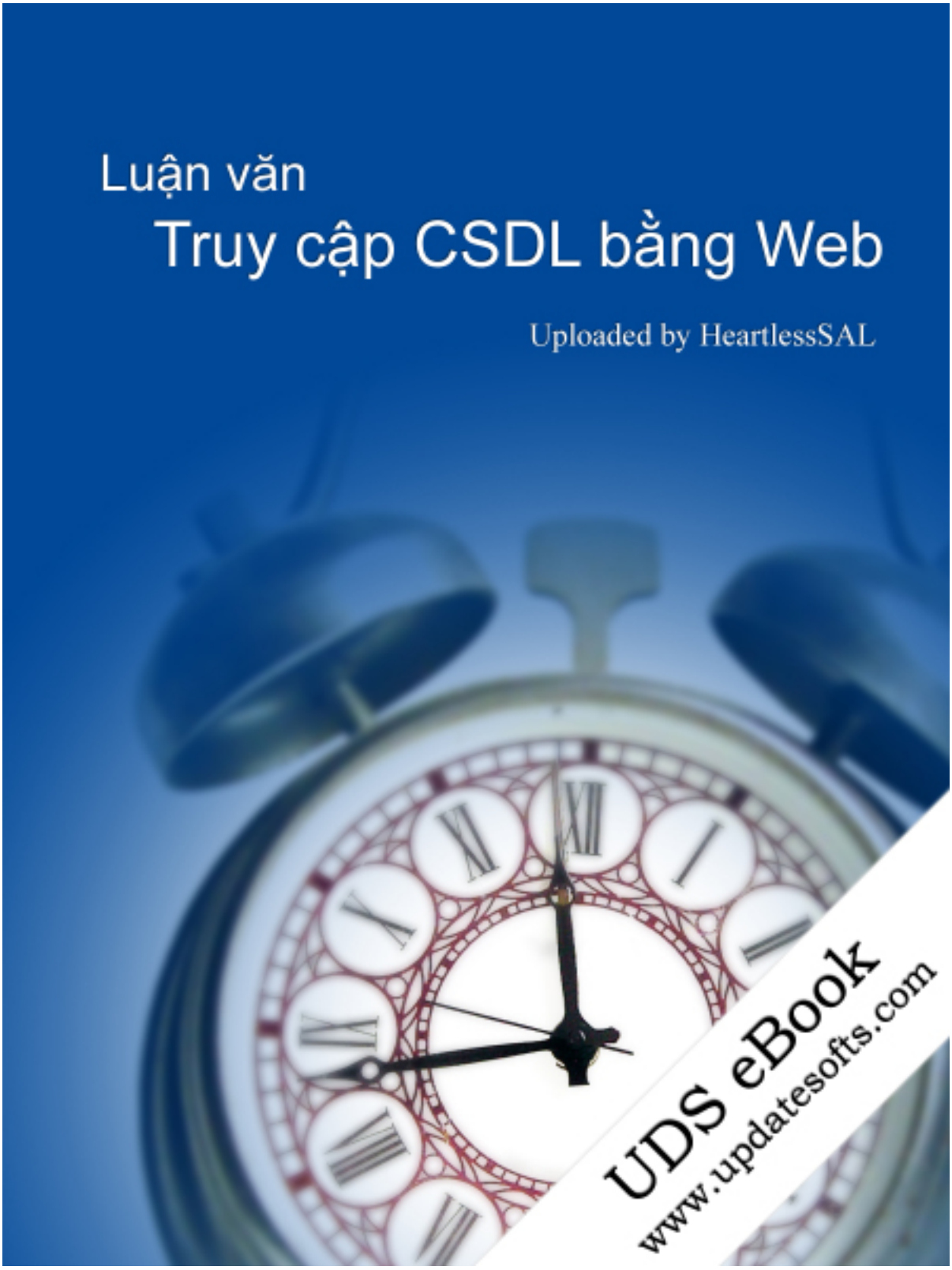
Luận văn

Truy cập CSDL bằng Web



MỤC LỤC

LỜI GIỚI THIỆU

Trong thời kỳ của kỷ nguyên thông tin hiện nay, vấn đề trao đổi thông tin là vô cùng quan trọng. Nhu cầu trao đổi thông tin gia tăng khi nền kinh tế ngày càng phát triển.

Do sự bùng nổ về thông tin như vậy người ta đã và đang rất quan tâm sử dụng công nghệ tin học đặc biệt là công nghệ Internet, Intranet. Các công nghệ này tạo điều kiện cho việc trao đổi và phổ biến thông tin dễ dàng không phụ thuộc vào vị trí địa lí. Công nghệ Internet, Intranet ban đầu chủ yếu phục vụ cho giáo dục, và nghiên cứu, nay đã mở rộng ra các lĩnh vực khác (thương mại, giải trí, ...).

Có rất nhiều phương pháp trao đổi thông tin (WWW, FTP-truyền file, EMAIL- thư điện tử, TELNET, RLOGIN - làm việc với máy tính từ xa, NEW-thảo luận, GOPHER - tìm kiếm file,...), trong đó dịch vụ WWW (World Wide Web) là một trong những dịch vụ được dùng phổ biến nhất.

Luận văn đi sâu nghiên cứu tìm hiểu dịch vụ World Wide Web trên mạng, và đặc biệt là tìm hiểu phương pháp khai thác cơ sở dữ liệu thông qua Web.

Luận văn được chia thành 2 phần:

Phần I:

Với tiêu đề ***Tổng quan về hệ thống Web***, phần này của luận văn trình bày những khái niệm cơ bản về Web, Web Client-Server. Trong phần này cũng trình bày những nội dung cơ bản nhất về ngôn ngữ HTML để xây dựng trang Web.

Phần II: Trình bày về các cách thức truy nhập cơ sở dữ liệu bằng chương trình ngoài, đặc biệt là qua giao diện CGI (Common Gateway Interface), các khái niệm cơ bản trong CGI. Trong phần này luận văn đưa ra hai phương thức truy nhập CSDL bằng CGI:

Truy nhập không hỗ trợ các công cụ của hệ quản trị cơ sở dữ liệu. Theo dạng này, luận văn chú trọng vào cách thức trao đổi thông tin theo dòng vào chuẩn và dòng ra chuẩn, luận văn phân tích sự hoạt động của chương trình CGI được viết trên C để thấy rõ cách thức trao đổi này.

Truy nhập nhờ công cụ hỗ trợ của hệ quản trị CSDL mà ở đây là hệ quản trị CSDL ORACLE. Luận văn trình bày hoạt động Oracle Web Server với các thuộc tính mở rộng so với các Web Server thông thường, các mở rộng này tạo điều kiện cho người phát triển xây dựng các ứng dụng với giao diện Web. Cuối cùng là xây dựng một chương trình ví dụ minh hoạ việc Oracle Web Server thao tác với cơ sở dữ liệu.

Em xin chân thành cảm ơn toàn thể các thày cô giáo Khoa CNTT, đặc biệt các thày giáo tổ bộ môn Các hệ thống thông tin, thày Hà Quang Thuỵ. Và các thày Trần Xuân Thuận, Lê Huy (Liên Hiệp Khoa học và sản xuất Phần Mềm - CSE) và toàn thể các anh chị ở CSE; những người đã cung cấp tài liệu, chỉnh sửa và đóng góp những ý kiến quý giá trong quá trình xây dựng luận văn này.

Hà Nội ngày 26-5-98

Tiêu Thị Dự
K39KTT-CNTT

PHẦN I TỔNG QUAN VỀ WEB

CHƯƠNG I HỆ THỐNG WEB

I Những khái niệm cơ bản về hệ thống Web

I.1 Nguồn gốc của World Wide Web

Năm 1990 nhóm nghiên cứu do Tim Berners-Lee đứng đầu làm việc tại phòng thí nghiệm vật lý hạt nhân châu Âu đã đưa ra một bộ giao thức mới phục vụ cho việc truyền và nhận các tệp siêu văn bản (Hypertext) trên mạng Internet. Bộ giao thức này chủ yếu dựa trên ngôn ngữ HTML (Hypertext Markup Language) để liên kết, trao đổi thông tin và gọi tắt là HTTP (Hypertext Tranfer Protocol). Ngay sau đó, các tổ chức và tập đoàn khác đã công nhận bộ giao thức HTTP, và thành lập một tổ chức gọi là W3 Consortium để tiếp tục phát triển và chuẩn hoá bộ giao thức này. W3 Consortium đã phát triển thêm các tính năng mới của HTML và các mức (Level) cũng như các chuẩn để thực hiện các phần mềm đi kèm. Từ đó thuật ngữ Worl Wide Web ra đời và được công bố rộng rãi trên Internet.

I.2 Khái niệm về Web

World Wide Web (viết tắt là WWW hay còn được gọi là Web) có cấu trúc thể hiện như một trang văn bản và đồ hoạ có các ***siêu liên kết***

(Hyperlinks) mà theo đó ta có thể lựa chọn. Những ***siêu liên kết*** này sẽ đưa chúng ta đến các tài nguyên khác trên mạng với đầy đủ tính năng độc đáo như các hình ảnh, đồ hoạ, âm thanh. . .Web giúp con người thực hiện những công việc trên mạng một cách dễ dàng.

I.3 Siêu liên kết

Siêu liên kết là một từ hay một cụm từ trên trang Web dùng để "chỉ" đến một trang Web khác. Khi nhấn chuột lên một siêu liên kết, trình duyệt sẽ đưa chúng ta đến một trang Web khác. Vì những liên kết siêu văn bản này thật sự là tính năng đặc trưng của Worl Wide Web, nên các trang Web thường được biết đến như là những tài liệu siêu văn bản.

I.4 Địa chỉ của Web

Địa chỉ của Web được biết đến nhờ các URL (Uniform Resource Locator -Bộ định vị tài nguyên đồng nhất). Nếu các trang Web được ghi lồng vào sâu hết mục này đến mục khác thì địa chỉ của Web sẽ hết sức dài. Nó thường được coi là một URL không bao giờ chấm dứt (liên kết với nhiều liên kết). Một URL thường có cấu trúc như sau:

Protocol://host.domain/directory/file.name

+ Protocol: Nghi thức TCP/IP sử dụng để tìm tài nguyên (HTTP hay FTP)
+ Host.domain : Tên máy chủ nơi trang Web lưu trú
+ Directory : Thư mục chủ chứa tài liệu đó
+ File.name : Tên chính xác của tập tài liệu đó

URL được sử dụng tất cả các dịch vụ thông tin trên mạng. Mỗi một trang Web có một URL duy nhất để xác định trang Web đó. Qua phân tích cấu trúc của một URL, ta thấy rằng thông qua URL có thể truy cập bất cứ một tài nguyên thông tin dữ liệu của bất kỳ dịch vụ nào của bất kỳ máy tính nào trên mạng.

I.5 Mô hình Web Client- Server

I.5.1 Web Browser

Web Browser là công cụ truy xuất dữ liệu trên mạng, là phần mềm giao diện trực tiếp với người sử dụng. Nó có khả năng yêu cầu thông tin từ Web Server và các dịch vụ khác nhau theo nhu cầu của người sử dụng. Sau đó Web Browser sẽ đợi thông tin từ Web Server hay các máy phục vụ của các dịch vụ thông tin khác và hiển thị thông tin cho người sử dụng. Thông tin hiển thị có thể được lưu trữ trên những trang Web riêng, được tạo ra trước khi có yêu cầu (đó là trang Web tĩnh) hoặc thông tin có thể được tạo ra từ trong các cơ sở dữ liệu dựa trên yêu cầu (đó là trang Web động). Có nhiều Web Browser khác nhau như:

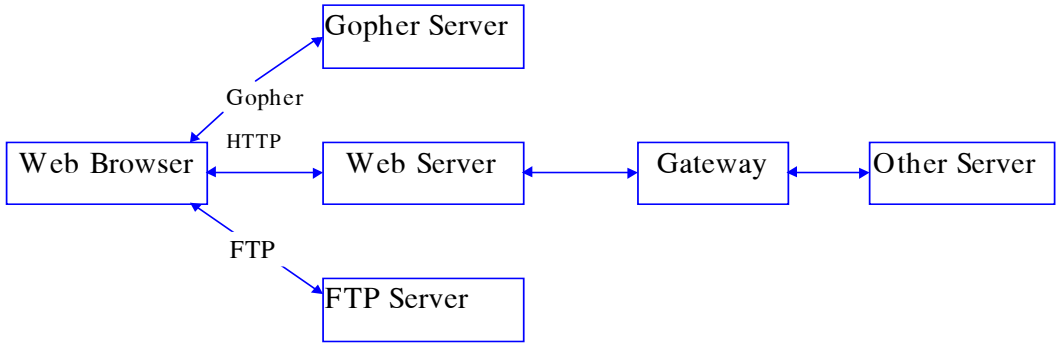
+ Mosaic
+ Netscape Navigator
+ Microsoft Internet Explorer
+ Lynx trong Unix

Phần lớn các Browser hỗ trợ một danh sách các đặc trưng với khả năng xử lý file như files HTML, Files GIF, Files JPEG. Nhiều Browser mới còn có khả năng hỗ trợ một danh sách các đặc trưng mở rộng có khả năng xử lý Java và JavaScpript. Nhiều Browser làm việc với file phụ thuộc vào Header kiểu MIME (Multipurpose Internet Mail Extentions). Các Browser như thế có thể tự xử lý files, và yêu cầu sự giúp đỡ của những ứng dụng, hay đơn giản là Save file vào đĩa.

I.5.2 Web Server

Web Server là một phần mềm đóng vai trò phục vụ. Khi được khởi động, nó được nạp vào bộ nhớ và đợi các yêu cầu từ nơi khác đến. Các yêu cầu có thể đến từ một người sử dụng dùng phần mềm Web Browser hoặc cũng có thể đến từ một Web Server khác. Trong cả hai trường hợp trên đối tượng

đưa ra yêu cầu gọi là khách hàng (Client). Các yêu cầu đối với Web Server thường là đòi hỏi về một tư liệu hay thông tin nào đó. Khi nhận được yêu cầu, nó phân tích để xác định xem tư liệu, thông tin khách hàng muốn là gì. Sau đó nó tìm lấy tư liệu và gửi cho khách hàng. Việc phục vụ phần lớn nhờ dịch vụ HTTP truy nhập đến tài liệu HTML hay những ứng dụng của CGI. Cũng có thể phục vụ thông qua các giao thức khác như: FTP, Gopher hay dịch vụ Telnet (minh hoạ hình 1.1)



Hình 1.1 Trao đổi thông tin Web Browser -

Nhiệm vụ chính của Web Server là:

+ "Tiếp nhận" yêu cầu đưa vào từ trên mạng

+ Cung cấp những trang HTML

+ Cung cấp và phát triển ứng dụng trên Web

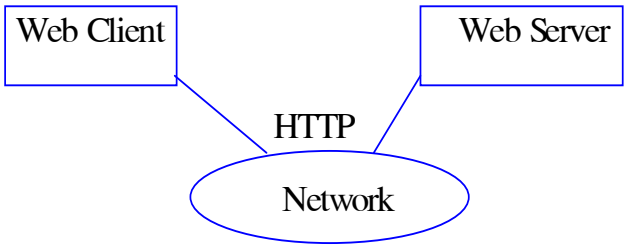
+ Tìm kiếm file từ một "Vitual root"

+ Phục vụ yêu cầu file tới Client

I.5.3 Web Client-Server

World Wide Web được xây dựng và hoạt động theo mô hình Client/Server. Các Client dùng một phần mềm gọi là Web Browser. Web Browser tiếp nhận thông tin yêu cầu từ người dùng sau đó gửi các yêu cầu tới máy Server xử lý.

Web Server cũng là một phần mềm chạy trên các máy phục vụ, nhận Request thực hiện theo yêu cầu rồi trả thông tin (Response) cho người sử dụng.



Hình 1.1 Web Client- Server Paradigm

I.6 Giao thức HTTP (Hypertext Transfer Protocol)

HTTP là giao thức truyền thông mà Client sử dụng để liên lạc với Server. Mọi giao thức truyền thông đều đòi hỏi một chương trình tương ứng trên Server để "nghe" yêu cầu đưa vào từ trên mạng. Ví dụ FPT có một FPT daemon, Telnet có một Telnet daemon giống như HTTP cũng có một HTTP daemon. Bởi vậy khi máy Server hoạt động đã có những daemon chạy trên Server, ví dụ như Web Listener trên Oracle Web Server cũng là một Server daemon tương ứng.

HTTP cũng tương tự như Telnet. Tuy nhiên có một sự khác biệt quan trọng giữa HTTP và Telnet đó là HTTP không duy trì kết nối với Server. Sau khi Server phục vụ một file tới Client, nó chấm dứt sự kết nối với trạm cuối. Trong thời gian sau đó nếu Client yêu cầu một file từ Server, thì khi một trang thông tin mới được tải xuống thì một kết nối mới mới được xây dựng với Server.

Không có một trạng thái thông tin nào có thể duy trì lâu dài giữa Client và Server và yêu cầu kết nối. Nếu thông tin đã yêu cầu mà Client phải bảo vệ nó hay nắm giữ nó, thông tin yêu cầu đó trình diện lại với kết nối sau.

I.7 Phân loại Web

Theo quan điểm của Martin Rennhackkawp (Tạp chí DBMS 5/97) cho rằng có hể phân loại Web thành 3 loại là: Trang Web tĩnh, Form page và trang Web động.

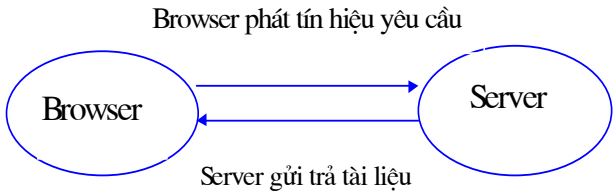
I.7.1 Trang Web tĩnh (Static Web Pages):

Trang Web tĩnh là tài liệu được phân phát rất đơn giản từ hệ thống file của Server. Phần mềm Web Server sẽ tiến hành tìm kiếm và xác định đúng vị trí file đó và gửi trả kết quả cho Client (Browser). Việc sử dụng trang Web tĩnh có những ưu, nhược điểm rõ ràng.

+ Ưu điểm: Khi cơ sở dữ liệu là nhỏ thì việc phân phát dữ liệu có hiệu quả, Server có thể đáp ứng nhu cầu của Client một cách nhanh chóng. Kiểu Web tĩnh sẽ là tốt nhất để sử dụng khi thông tin có sẵn trên ổ đĩa cứng, và không thay đổi.

+ Nhược điểm: Không năng động, không đáp ứng nhu cầu thông tin vì vậy không đáp ứng được những yêu cầu phức tạp của người sử dụng.

Quá trình phân phát tài liệu tĩnh được thể hiện như sau:



Hình I.2 Phân Phát tài liệu

I.7.2 Form Pages

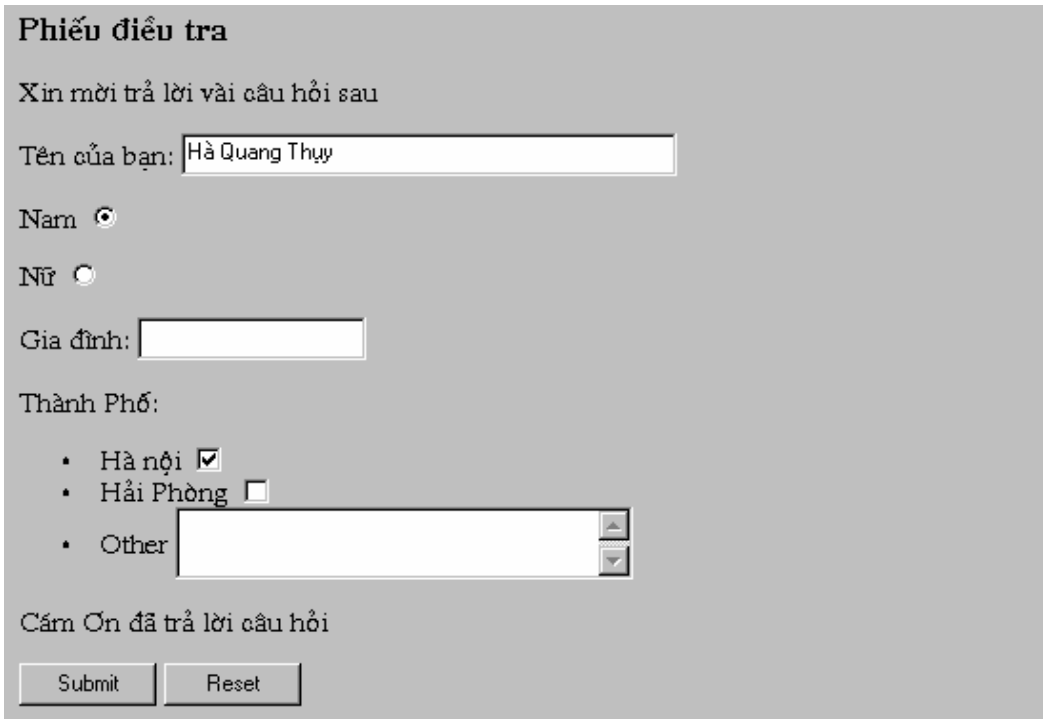
Về mặt bản chất Form Pages là trường hợp đặc biệt của trang Web tĩnh. Nó cho phép nhận được phản hồi từ phía người sử dụng thông qua form. Form pages được xây dựng dựa trên ngôn ngữ HTML.

Ví dụ

```
<HTML>
<HEAD>
<title>Ví dụ về form  pages</title>
</HEAD>
<BODY>
<H3>Phiếu điều tra</H3>
<P>Xin mời ngài trả lời vài câu hỏi sau</p>
<Form Method="POST" ACTION="HTTP://www.hal.com/Sample">
<P>Tên của bạn: <INPUT Name="name" size="48">
<P>Nam <INPUT Name="gender" TYPE=RADIO VALUE="Nam">
<P>Nữ <INPUT Name="gender" TYPE=RADIO VALUE="Nữ">
<p>Gia đình: <INPUT Name="Family" TYPE=Text>
<p>Thành Phố:
<UL>
```

<LI>Hà nội <INPUT Name="city" TYPE="Checkbox VALUE="Hà nội"
<LI>Hải Phòng <INPUT Name="city" TYPE="Checkbox VALUE="Hải Phòng"
<LI>Hồ Chí Minh<INPUT Name="city" TYPE="Checkbox VALUE="Hồ Chí Minyyy<LI>Other <TEXTAREA Name="Other" Cols=48 rows=4</TEXTAREA>
</UL>
<P>Cám Ơn Bạn đã trả lời câu hỏi</p>
<P><INPUT TYPE=SUBMIT> <INPUT TYPE=RESET>
</Form>
</BODY>
</HTML>

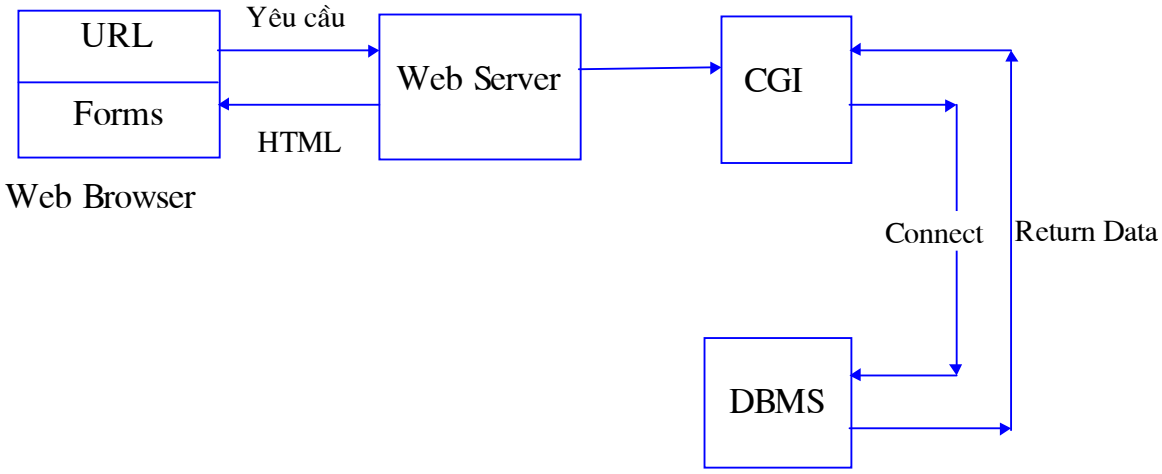
Sau khi trình duyệt Web thực hiện, nhận được kết quả như sau:



Trong ví dụ trên, các thành phần <P> và <UL> được dùng để bố trí các trường ký tự và nhận số liệu. Nhiều bộ duyệt quy ước dùng các phím Tab và Shift để chuyển dịch giữa các trường và ENTER để SUBMIT, nghĩa là kết thúc quá trình vào số liệu và gửi đi. Nút SUBMIT dùng để E_Mail hay gửi thẳng nội dung của bảng đến Server, tuỳ thuộc vào thuộc tính ACTION. Nút RESET trả các trường nhận về giá trị ban đầu.

I.7.3 Web động:

Trang Web loại này có thể thao tác với cơ sở dữ liệu để đáp ứng nhu cầu phức tạp của người sử dụng. Chẳng hạn như khi người sử dụng cần có những thông tin thay đổi hàng ngày thì việc phải thao tác với cơ sở dữ liệu bên ngoài là cần thiết. Có nhiều cách thức có thể truy nhập đến cơ sở dữ liệu bên ngoài ví như ISAPI (Internet Server Application Programming Interface), ASP (Active Server Pages) hay JAVA và điển hình là dùng chương trình chạy ngoài CGI (Common GateWay Interface -Sẽ được trình bày kỹ phần sau). Cơ chế hoạt động được thể hiện như hình vẽ1.3:



Hình I.3 Cơ chế hoạt động của Web Server

Khi Client gửi yêu cầu tới Server thông qua CGI, chương trình CGI sẽ

móc nối với Cơ sở dữ liệu bên ngoài, thực hiện chương trình sau đó gửi trả kết quả dưới dạng HTML và hiển thị trên trang Web.

Nhưng việc thực hiện chương trình CGI có vấn đề về thời gian tức là việc tải files sẽ diễn ra chậm bởi vậy người ta đưa ra giải pháp khắc phục là dùng phần mềm trung gian (MiddleWare) - **ODBC** (Open Database Connectivity).

ODBC là một chương trình ứng dụng chuẩn để truy nhập dữ liệu. Phần mềm ODBC có chức năng ***kết nối*** với cơ sở dữ liệu (Connection Managenal), và do chỉ hiểu được câu lệnh SQL nên nó còn giữ vai trò ***thông dịch.*** Việc dùng ODBC cũng có những ưu nhược điểm như sau:

+ Ưu điểm: khắc phục được tình trạng quá tải trên Web Server và có thể làm việc được với nhiều cơ sở dữ liệu cùng một thời điểm, tốt đối với mạng LAN. Và do được ra đời khá sớm và quen thuộc nên các công ty sản xuất máy tĩnh cũng chú ý hỗ trợ ODBC .

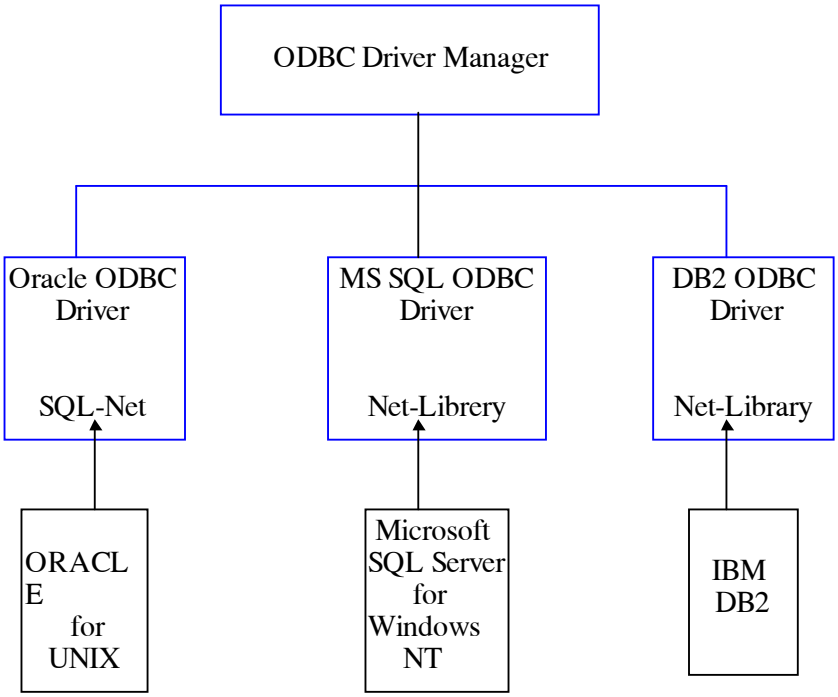
+ Nhược điểm: Phải cài đặt từng ODBC Driver tương ứng với phần mềm CSDL trên Clients nếu muốn chạy CSDL đó. Mặt khác ODBC lại không phù hợp đối với mạng diện rộng.

Hình I.4 thể hiện cách thức ODBC trong đó:

+ ODBC Manager : Gửi đi các cuộc gọi từ những ứng dụng khác đến những thiết bị chuẩn .

+ Driver: Xử lý các chức năng ODBC, trình diện câu lệnh SQL tới BDMS xác định, và trả lại kết quả.

+ DBMSs: Xử lý yêu cầu từ ODBC Driver và trả lại kết quả.



Hình I.4 Cách thức ODBC

CHƯƠNG II NGÔN NGỮ SIÊU VĂN BẢN (HTML)

I Khái niệm chung

Ngôn ngữ siêu văn bản HTML (Hypertext Markup Langure) là một cách đưa vào văn bản nhiều thuộc tính cần thiết để có thể truyền thông quảng

bá trên mạng toàn cầu WWW (World Wide Web). HTML cho phép đưa hình ảnh đồ hoạ vào văn bản, và tạo những tài liệu siêu văn bản có khả năng đối thoại tương tác với người dùng.

HTML chủ yếu xoay quanh khái niệm "tiêu thức" (tag) làm nền tảng. Để tạo một siêu văn bản, ta có thể dùng bất cứ một chương trình soạn thảo nào ví dụ như NC, EDIT của DOS, NotePad hay Write của Windows 3.x, WordPad của Win95. . . Và chỉ cần nắm vững các tiêu thức của HTML, và chú ý khi cất lên đĩa thì cần lưu dưới dạng TXT. Song có một hạn chế là dạng văn bản khi soạn với khi xem sau này trên WWW là không giống nhau.

Ngày nay do sự phát triển của mạng toàn cầu, HTML cũng ngày càng trở nên phức tạp và hoàn thiện hơn để đáp ứng được những yêu cầu mới nảy sinh trong quá trình phát triển đó (như âm thanh, hình ảnh động, hay điều khiển từ xa, hiện thực ảo. . .). Người ta gọi đó là những phiên bản của HTML và đánh số để biểu thị.

Một trong những điểm mạnh của HTML là một văn bản bất kỳ nếu tuân thủ tiêu chuẩn HTML đều có thể hiện được lên màn hình hay in ra, tóm lại là hiểu được, bởi bất kỳ loại phần mềm hay máy tính nào mà người dùng có, không phân biệt Netscape trên Windows, hay Lynx trên Unix, thậm chí cho người khiếm thị bằng phần mềm đặc biệt.

II Đặc tả về HTML

Toàn bộ các thẻ của HTML được chia ra thành 7 nhóm thành phần như sau và được gọi là từ khoá :

- Từ khoá xác lập cấu trúc tài liệu
- Từ khoá tạo điểm móc nối
- Từ khoá định dạng khối

- Từ khoá khai báo danh sách
- Từ khoá khai báo loại thông tin và định dạng mẫu chữ
- Từ khoá đưa hình ảnh vào tài liệu
- Từ khoá lập mẫu biểu bảng

II.1 Các từ khoá định dạng cấu trúc tài liệu

Các thành phần xác định cấu trúc tài liệu là bắt buộc phải có trong tài liệu HTML. Ngoài phần mở đầu xác định tên và một số thuộc tính để phân biệt giữa các tài liệu, chỉ có những thành phần sau là bắt buộc phải có trong một tài liệu HTML để phù hợp với chuẩn. Sau đây là cấu trúc cơ sở của trang Web được xây dựng bằng HTML Những từ khoá thiết yếu đó và trình tự xuất hiện của chúng được sơ bộ liệt kê như sau:

<HTML>
<HEAD>
<TITLE>. . .</TITLE>
</HEAD>
<BODY>
. . . . . .
</BODY>
</HTML>

- <HTML>. . . </HTML>

Cặp từ khoá này giúp nhận dạng tài liệu có chứa các thành phần tuân thủ theo chuẩn về ngôn ngữ HTML

- <HEAD>. . .</HEAD>

Thành phần mở đầu của một tài liệu HTML chứa các thông tin về tài liệu đó. Trong đó cặp từ khoá đặt tiêu đề cũng là bắt buộc:

<HEAD>
<TITLE>Giới thiệu chung về trang Web</TITLE>
</HEAD>
Cặp từ khoá <HEAD> và</HEAD> không trực tiếp ảnh hưởng đến cách thể hiện tài liệu khi ta xem bộ duyệt.

Các thành phần sau đây liên quan tới thành phần mở đầu tài liệu tuy không trực tiếp "tạo dáng" nhưng nếu sử dụng lại cung cấp những thông tin quan trọng đối với bộ duyệt:

<BASE> Cho phép khai báo địa chỉ cơ sở của tài liệu
<ISINDEX> Cho phép tìm kiếm trong tài liệu theo từ khoá
<LINK> Chỉ ra mối quan hệ giữa các tài liệu
<NEXTID> Tạo tên gọi đồng nhất hoá tài liệu
<META>Cung cấp thông tin hữu ích cho chế độ Server/Client

- Thẻ <BODY>. . .</BODY>:

Phần thân của trang Web chứa tất cả các thành phần khác cũng như nội dung từ lời văn đến hình ảnh cấu thành một tài liệu, song không dính dáng gì đến sự bài trí của tài liệu đó.

II.2 Điểm móc nối <A>. . .</A>

Đánh dấu cụm từ chỉ đến một kết nối siêu văn bản (Hypertext link) mà khi trỏ tới nó, bộ duyệt sẽ dẫn dắt đến một tài liệu hoặc một đoạn văn khác. Có nhiều thuộc tính nhưng hoặc NAME hoặc HREF là thuộc tính bắt buộc.

- HREF

Nếu có thuộc tính HREF, cụm từ đứng giữa sẽ trở thành siêu văn bản, nghĩa là nó trỏ đến một văn bản khác chứ không chỉ mang nội dung thuần tuý. Khi chọn vào cụm từ đó, một tài liệu khác hoặc một đoạn tài liệu khác

trong cùng tài liệu đang xem mà địa chỉ được chỉ ra bởi thuộc tính HREF sẽ được hiện lên.

- NAME

Dùng để đặt tên cho điểm móc nối và vì vậy phải là duy nhất trong nội bộ tài liệu hiện thời mặc dù tên có thể đặt một cách tuỳ ý

Ví dụ:
<A Name=coffee>Cà phê</A> là một ví dụ về loại cây
Từ tài liệu khác có thể tham chiếu tới bằng cách đặt tên gọi vào sau địa chỉ, ngăn cách bằng một dấu #.

- TITLE

Thuộc tính này chỉ có ý nghĩa thông báo và được dùng để đặt đầu đề cho tài liệu mà địa chỉ đó do HREF chỉ ra. Đầu đề cần phải là duy nhất đối với tài liệu đích.

Bộ duyệt có thể hiện đầu đề của tài liệu trước khi lấy về, chẳng hạn như một ghi chú nhỏ bên lề hay trong một khung nhỏ khi con trỏ chuột di qua điểm móc nối (có thể là một cụm từ hay một hình ảnh), hay khi đang tải tài liệu ra (nhất là khi qua đường truyền có tốc độ không cao lắm).

Có những tài liệu không có đầu đề như đồ hoạ, thực đơn Gopher,. . .

II.3 Các từ khoá định dạng khối

Các thành phần định dạng khối dùng để định dạng cả một đoạn văn bản và phải nằm trong phần thân của tài liệu. Có những cặp từ khoá quan trọng sau đây:

- <ADDRESS>. . .</ADDRESS> Định dạng phần địa chỉ
- <Hn>. . .</Hn>(n là chữ số từ 1 đến 6) Định dạng sáu mức tiêu đề.

HTML có 6 mức tiêu đề bao hàm kiểu phông chữ, cách đoạn trước sau cũng như khoảng trống cần thiết để thể hiện tiêu đề. Mức cao nhất là <H1>, kế đến là<H2>...cho đến <H6>.

Cách thể hiện phụ thuộc vào bộ duyệt, nhưng thông thường thì:

<H1>...</H1> Chữ đậm, cỡ lớn, căn giữa. Một, hai dòng cách trên và dưới.

<H2>...</H2> Chữ đậm, cỡ lớn căn lề trái. Một, hai dòng cách trên và dưới.

<H3>...</H3> Chữ nghiêng, cỡ lớn, căn lề trái, hơi lùi vào trong. Một hay hai dòng cách trên và dưới.

<H4>...</H4> Chữ đậm, cỡ thường lùi vào trong nhiều hơn H3. Một dòng cách trên và dưới.

<H5>...</H5> Chữ nghiêng, cỡ thường, lùi vào trong nhiều hơn H4. Một dòng cách trên.

<H6>...</H6> Chữ đậm, cỡ thường, lùi vào trong nhiều hơn H5. Một dòng cách trên.

- <HR> Đường phân cách ngang tài liệu
- <P>. . .</P> Giới hạn một Paragraph

Chỉ là giới hạn một Paragraph. Cách bài trí do các thành phần khác tạo thành. Thường có khoảng trống khoảng một dòng hay nửa dòng trước paragraph, trừ khi nằm trong phần địa chỉ. Một số bộ duyệt thể hiện dòng đầu của Paragraph tụt vào .

-
 Bẻ dòng

Bắt buộc xuống dòng tại vị trí gặp từ khoá này. Dòng mới được căn lề như dòng được kẻ tự động khi dòng đó quá dài.

- <PRE>. . .</PRE> Đoạn văn bản đã định dạng sẵn

Giới hạn đoạn văn bản đã được định dạng sẵn cần được thể hiện bằng phông chữ có độ rộng ký tự không đổi. Nếu không có thuộc tính WIDTH đi cùng thì bề rộng mặc định là 80 ký tự/dòng. Bề rộng 40,80,132 được thể hiện tối ưu, còn các bề rộng khác có thể được làm tròn trong thành phần định dạng trước:

* Dấu xuống dòng sẽ có ý nghĩa chuyển sang dòng mới (chứ không còn là dấu cách)
* không dùng nếu có sẽ được coi như là xuống dòng.
* Được phép dùng các thành phần liên kết và nhấn mạnh.
* Không được chứa các thành phần định dạng paragraph (tiêu đề, địa chỉ).
* Ký tự TAB phải hiểu là số dấu cách nhỏ nhất sao cho đến ký tự tiếp theo ở vị trí là bội của 8. Tuy nhiên không nên dùng.

Ví dụ:

<PRE WIDTH=”50”>
Nguyễn văn Trỗi - Cử nhân.
Nguyễn viết Xuân - Kỹ sư.
</PRE>

- <BLOCKQUOTE>. . .</BLOCKQUOTE> Trích dẫn nguồn tài liệu khác

Dùng để trích dẫn một đoạn văn bản,thường được thể hiện bằng chữ nghiêng có căn lề lùi vào trong và thường có một dòng trống ở trên và dưới.

II.4 Các từ khoá khai báo danh sách

HTML hỗ trợ nhiều kiểu loại danh sách, tất cả đều có thể lồng vào nhau và chỉ nên dùng trong phần thân của tài liệu (<BODY>. . .</BODY>).

- <DL>. . .</DL> Danh sách định nghĩa

Dùng để lập danh sách các thuật ngữ và định nghĩa tương ứng

ví dụ

<DL COMPACT>

<DT> Mèo <DD> Là một loại động vật

<DT>Hoa Ngọc Lan<DD> Là một loài thực vật

</DL>

Trong ví dụ trên thì:

<DT> Chỉ tên thuật ngữ

<DD> Chỉ phần định nghĩa. Có thể có thêm thuộc tính COMPACT để chỉ dẫn thêm là xếp <DT> và <DD> theo từng cặp. Lúc đó sẽ phải viết là <DL COMPACT> và tiếp theo là <DT>.

- <DIR>. . .</DIR> Danh sách kiểu thư mục

Danh sách các phần tử mà trong đó mỗi phần tử dài đến khoảng 20 ký tự. Sau <DIR> bắt buộc phải là <LI> (List Item)

- <MENU>. . .</MENU> Danh sách kiểu thực đơn

Danh sách các lựa chọn trong một thực đơn. Sau <MENU> phải là <LI>.

ví dụ

<MENU>

<LI> Con mèo

<LI> Con mèo con

</MENU>

- <OL>. . .</OL> Danh sách có sắp xếp

Danh sách có sắp xếp theo trình tự hay mức độ quan trọng. Sau <OL> phải là <LI> và có thể thêm thuộc tính COMPACT .

- <UL>. . .</UL> Danh sách không có sắp xếp

Giống <OL> nhưng danh sách không được sắp xếp

II.5 Các từ khoá khai báo loại thông tin và định dạng mẫu chữ

II.5.1 Khai báo loại thông tin

Có những cặp từ khoá tuy khác nhau nhưng lại thể hiện như nhau, cụ thể có những cặp từ khoá khai báo loại thông tin như sau:

* <CITE> . . .</CITE> Trích dẫn
* <CODE> . . .</CODE> Ví dụ về mã lệnh
* <EM>. . .</EM> Nhấn mạnh
* <KBR>. . .</KBR> Ký tự do người dùng gõ vào trên bàn phím
* <SAMP>. . .</SAMP> Nguyên văn
* <STRONG>. . .</STRONG> Rất nhấn mạnh
* <VAR>. . .</VAR> Chỉ tên biến hay tham số

II.5.2 Định dạng mẫu ký tự

* <B>. . .</B> Thể hiện chữ đậm
* <I>. . .</I> Chữ nghiêng
* <TT>. . .</TT> Chữ đánh trên máy chữ

II.6 Lồng hình ảnh <IMG. . .>

Dùng để lồng hình ảnh vào tài liệu. Không dùng để chèn siêu văn bản khác. Có thể thêm những thuộc tính sau

- ALIGN

Căn lề trên (TOP), giữa (MIDLE) hay dưới (BOTTOM), các ký tự văn bản đối với hình ảnh.

- ALT

Dùng trong trường hợp có thể bộ duyệt không hiển thị được hình ảnh phải hiện dòng văn bản thay thế.

- ISMAP

Hình ảnh có dạng một bản đồ, nghĩa là có thể chứa các vùng được "ánh xạ" đến những URL và khi bấm vào những vị trí khác nhau trên hình vẽ đưa đến tài liệu khác nhau.

- SRC

Đây là thuộc tình bắt buộc, với gía trị là một URL của hình ảnh được lồng vào. Cú pháp cũng như ở HREF trong thành phần liên kết<A>.

II.7 Các từ khóa lập mẫu biểu bảng (Forms)

Từ HTML 2.0 trở đi, các từ khoá lập bảng biểu được đưa vào cho phép nhận được phản hồi từ phía người dùng, bằng cách đặt những trường input (vào số liệu) bên cạnh những thành phần khác, cho phép có độ linh hoạt rất đáng kể trong thiết kế bảng biểu:

- <FORM>. . .</FORM> Giới hạn một bảng

Có thể có nhiều bảng trong một tài liệu song thành phần này không được phép lồng nhau. Thuộc tính ACTION là một URL cho biết nơi mà nội dung của bảng được gửi đến để xử lý, mặc định là URL của tài liệu hiện thời nếu không có thuộc tính này đi kèm. Phương thức gửi tuỳ thuộc vào giao thức truy nhập mà URL chỉ ra cũng như giá trị của các thuộc tính METHOD và ENCTYPE. Một cách tổng thể:

* METHOD dùng để chọn phương thức
* ENCTYPE định dạng khuôn của số liệu trong trường hợp giao thức không bao hàm luôn chính khuôn dạng ấy.

Nếu thuộc tính ACTION là một URL với giao thức HTTP, thuộc tính METHOD phải chứa một phương thức HTTP theo tiêu chuẩn IETF. Mặc định của METHOD là GET. Mặc dù trong nhiều trường hợp, phương thức POST

được ưa chuộng hơn. Với phương thức POST, thuộc tính ENCTYPE là một kiểu MINE cho biết khuôn dạng của số liệu đưa đến, mặc định là application/x/_www_form_rlencoded (tên ứng dụng/x_www_bảng mã hoá theo run length). Trong bất kỳ tình huống nào thì về mặt logic, nội dung của bảng mã sẽ gồm những cặp tên gọi/giá trị. Tên gọi thường trùng với giá trị của thuộc tính NAME.

- <INPUT>. . .</INPUT> Giới hạn một trường Input

Dùng để khai báo một trường mà người dùng đưa số liệu vào. Gồm có các thuộc tính như sau:

* ALIGN: Các giá trị cho phép hoàn toàn giống như thuộc tính ALIGN của thành phần <IMG. . .>

* CHECKED:

Để chỉ một nút chọn kiểu đánh dấu hay kiểu nút Radio.

* MAXLENGTH:

Số ký tự tối đa có thể gõ vào một trường (mặc định là vô hạn), được phép lớn hơn SIZE và khi đó trường này sẽ được cuộn.

* NAME:

Thuộc tính hay dùng nhất để chỉ tên gọi tượng trưng, dùng khi truyền đi nội dung.

* SIZE

Khai báo kích thước hay độ chính xác của một trường tuỳ theo kiểu của nó. Ví dụ để khai báo một trường rộng 24 ký tự thì ta khai báo như sau:

INPUT TYPE =text SIZE=”24”

* SRC

Một URL của hình ảnh, chỉ dùng với TYPE=IMAGE trong HTML 2.0.

* TYPE

Khai báo kiểu số liệu (ngầm định là ký tự), với các kiểu sau:

+ CHECKBOX

Dùng cho kiểu logic Bool với giá trị mặc định là 'on'

+ HIDDEN

Không hiện lên đối với người sử dụng nhưng vẫn được gửi cùng với nội dung của bảng.

+ IMAGE

Một trường mang hình ảnh mà khi bấm vào đó, bảng sẽ được trình và các toạ độ của điểm chọn tính bằng pixel tính từ góc trái trên của ảnh cùng các cặp tên /trị khác, trong đó toạ độ x thì tên của trường có thêm .x, toạ độ y thêm .y vào. Mọi thuộc tính VALUE đều bị bỏ qua. Còn chính hình ảnh thì do thuộc tính SRC chỉ ra.

+ PASSWORD:

Cũng giống như TEXT có điều ký tự không hiện lên khi người dùng gõ vào (như khi vào mật khẩu).

+ RADIO:

Để nhận một giá trị trong số các giá trị có thể có, và đòi hỏi phải có VALUE đi kèm.

+ RESET: Là nút mà khi bấm vào sẽ đặt các trường Input về các giá trị ban đầu. Nhãn của nút được khai báo như ở nút SUBMIT.

+ SUBMIT: Là nút mà khi bấm vào sẽ kết thúc quá trình vào số liệu và bảng sẽ được gửi đi. Thuộc tính VALUE cho phép gán nhãn cho nút. Nếu thuộc tính NAME cũng có thì một cặp tên/trị cũng được gửi đi.

+ TEXT: Dùng để vào một dòng kí tự, thường đi cùng với SIZE và MAXLENGTH.

+ VALUE: Dùng để khai báo giá trị ban đầu (cũng có nghĩa là mặc định) của trường kí tự hay số, hoặc giá trị trả lại khi được chọn đối với trường logic Bool. Thuộc tính này bắt buộc đối với trường thuộc tính RADIO.

- <SELECT>. . .</SELECT> Một thành phần lựa chọn thì bao gồm nhiều tuỳ chọn.

Cho phép người dùng chọn từ một danh sách mà từng phần tử được khai báo bằng <OPTION> với các thuộc tính sau:

* MULTIPLE: Cho phép chọn nhiều phần tử cùng một lúc (<SELECT MULTIP>).

* NAME: Khai báo tên biến tương ứng

* SIZE: Khai báo số phần tử hiện lên, nếu >1 thì hộp đối thoại sẽ là một danh sách.

Thành phần SELECT thường được thể hiện dưới dạng một danh sách kéo xuống hoặc kéo lên. Nếu không có OPTION nào có SELECTD thì phần tử đầu tiên trong danh sách sẽ mặc định là được chọn.

Ví dụ

```
<SELECT NAME=”Tự nhiên”>
<OPTION>Toán
<OPTION>Lý
<OPTION>Hoá
</SELECT>
```

- <OPTION> Một tuỳ chọn trong một thành phần lựa chọn

Chỉ xuất hiện trong một thành phần <SELECT>. . .</SELECT>, biểu thị một tuỳ chọn và có thể có các thuộc tính sau:

* DISABLED: Chỉ ra rằng chưa dùng đến trong bảng này mà sẽ dùng trong tương lai.

* SELECTED: Chỉ ra rằng tuỳ chọn này là mặc định.

* VALUE: Nếu có sẽ cho biết giá trị trả lại nếu được chọn, mặc định là là nội dung của chính thành phần này (những gì mà chính người sử dụng nhìn thấy).

<TEXTAREA >. . .</TEXTAREA> Một trường nhận số liệu nhiều dòng, cho phép người dùng nhập nhiều hơn một dòng ký tự

Ví dụ

<TEXTAREA NAME=”address” ROWS=64 COLS=6>
Liên hiệp sản xuất phần mềm (CSE)
Số 21-Lý Nam Đế-Hà Nội
</TEXTAREA>

Đoạn nằm giữa <TEXTAREA> và </TEXTAREA> được dùng làm giá trị ban đầu cho trường này. <TEXTAREA> là bắt buộc kể cả khi giá trị ban đầu đó không có. Khi chuyển đi các dòng cần kết thúc bằng CR/CL. Các thuộc tính ROWS và COLS để khai báo kích thước dòng và cột của cửa sổ số liệu, nếu số liệu nhiều hơn thì bộ duyệt sẽ cho phép cuộn (scroll).

II.8 Lập bảng

- <TABLE>...</TABLE> Giới hạn bảng

Mặc định bảng không có đường bao nếu không đi với thuộc tính BORDER. Có các thuộc tính sau:

* <BORDER> Để định nghĩa một bảng có đường bao, không làm ảnh hưởng đến độ rộng của bảng.

BORDER=”giá trị” có các giá trị xác định (0,1,2...) tượng trưng cho các màu của đường bao nếu cho giá trị =0 thì đường bao có màu giống màu màn hình. Cho phép điều chỉnh độ dày của đường bao ngoài đậm hơn đường bao trong để dễ nhìn hơn.

* CELLSPACING=<giá trị> Các giá trị là khoảng cách giữa các ô
* CELLPADING=<gía trị> Các giá trị xác định khoảng cách giữa nội dung của ô và đường bao.

Ví dụ

<TABLE BORDER=0 CELLSPACING=0 CELLPADDING=0> cho ta bảng Compact nhất có thể có (dành tất cả cho nội dung).

* WIDTH=<giá trị hay phần trăm>

Nếu đi cùng với TABLE thuộc tính này có ý nghĩa mô tả chiều rộng mong muốn của bảng. Thường thì bộ duyệt tự tính toán sao cho bảng được bố trí hợp lý. Dùng thuộc tính này bắt buộc bộ duyệt phải cố gắng làm sao xếp được các ô vào bảng có độ rộng mong muốn đó.

Nếu đi cùng với <TH> hay <TR> lại có ý nghĩa tương tự đối với một ô.

- <TR>. . .</TR> Định nghĩa một hàng

Khai báo một hàng của bảng, với thuộc tính ALIGHN và VALIGN để chỉ ra cách căn lề của nội dung trong các ô của hàng hiện thời.

* ROWSPAN: Cho biết ô hiện thời chiếm mấy hàng của bảng, mặc định là 1.
* ALIGN

Nếu nằm trong <CAPTION> nó có ý nghĩa đối với đầu đề của bảng nằm trên hay nằm dưới, có giá trị hoặc TOP hoặc BOTTOM ngầm định là TOP. Nếu nằm trong <TR>, <TH>, hay <TD> có thể có các giá trị LEFT,

CENTER hay RIGHT và điều khiển việc đặt nội dung của ô căn bên trái hay vào giữa, hay ở ô bên phải.

* VALIGN

Nằm trong <TR>, <TH> hay <TD> có thể có các giá trị TOP, MIDDLE, BOTTOM hay BASELINE để điều khiển việc đặt nội dung của ô lên trên, (vào giữa theo chiều dọc) hay xuống dưới và cũng có thể tất cả là các ô trong hàng cùng căn theo một đường nằm nghang

- <TD>. . .</TD> Định nghĩa một ô

Có nghĩa là dữ liệu bảng (Table Data), chỉ được xuất hiện trong cùng một hàng của bảng. Mỗi hàng không nhất thiết phải có cùng số ô vì dòng ngắn hơn sẽ được chắp thên ô rỗng vào bên phải. Mỗi ô chỉ được chứa các thành phần bình thường khác nằm trong phần thân của tài liệu. Các thuộc tình mặc định là:

ALIGN=left và VALIGN=middle.

Các mặc định này có thể thay đổi bởi các thuộc tính trong <TR> và lại thay đổi tiếp bởi thuộc tính ALIGN hoặc VALIGN khai báo riêng cho từng ô. Bình thường mặc định thì nội dung sẽ được "bẻ dòng" cho vừa vào khổ rộng của từng ô. Dùng thuộc tính NOWRAP trong <TD> để cấm việc đó. <TD >. . .</TD> cũng có thể chứa các thuộc tính NOWRAP, COLSPAN và ROWSPAN.

+ NOWRAP: Khi dùng thuộc tính này đề phòng ô quá rộng.

+ COLSPAN: Có thể xuất hiện trong bất kỳ ô nào (<TH> hay<TD>) và chỉ ra rằng ô đó chiếm mấy ô của bảng, mặc định là 1.

- <TH>. . .</TH> Ô chứa tiêu đề

Có nghĩa là tiêu đề của bảng (Table Header), các ô này tương tự như các ô bình thường khác được định nghĩa bằng <TD>, có điều phông chữ là đậm và có thuộc tính mặc định là ALIGN=Center.

<TH. . .>. . .</TH> cũng có thể chứa thuộc tính VALIGN, NOWRAP, COLSPAN và ROWSPAN.

- <CAPTION>. . .</CAPTION> Đầu đề của bảng.

Đặt đầu đề cho một bảng nên phải nằm trong một cặp <TABLE> song không được nằm trong hàng hay cột. Thuộc tính mặc định là ALIGN=Top (đầu đề đặt ở đầu bảng), song có thể đặt là ALIGN =Bottom (cuối bảng). Đầu đề có thể chứa bất kỳ thành phần nào một ô có thể chứa và luôn được căn lề vào giữa bảng (theo chiều ngang) và có thể cũng có bẻ dòng cho phù hợp.

Ví dụ ta có thể tạo được bảng gồm 2 dòng 3 cột như sau

```
<TABLE  BORDER>
<TR>
<TD>A</TD> <TD>B</TD> <TD> C</TD>
</TR>
<TR>
<TD>D</TD >  <TD>E</TD> <TD>F</TD>
</TR>
</TABLE>
```

A	B	C
D	E	F

Đặc biệt ô rỗng thì không có đường bao, muốn ô rỗng có đường bao phải dùng dấu cách không bẻ dòng. Có thể lợi dụng ROWSPAN và COLSPAN để tạo bảng có ô chồng chéo lên nhau nhưng không lên lạm dụng.

III Các ưu nhược điểm của HTML

- Ưu điểm: Ngôn ngữ HTML có ưu điểm so với nhiều loại ngôn ngữ khác đó là:

  + Dễ đọc, dễ hiểu, dễ sử dụng

  + Không phụ thuộc vào Hệ điều hành

  + Giảm thông lượng đường truyền

  + Liên kết nhiều dạng thông tin và các dịch vụ thông tin khác trên Mạng

- Nhược điểm: Ngôn ngữ HTML có một số nhược điểm cơ bản sau

  + Là ngôn ngữ thông dịch, do đó nó sẽ giảm tốc độ thực hiện của các ứng dụng trên Web

  + Khó đảm bảo về an toàn và bảo mật

  + Không hỗ trợ đa ngôn ngữ.

PHẦN II

TRUY NHẬP CƠ SỞ DỮ LIỆU THEO GIAO DIỆN CGI

CHƯƠNG I GIỚI THIỆU CHƯƠNG TRÌNH CGI

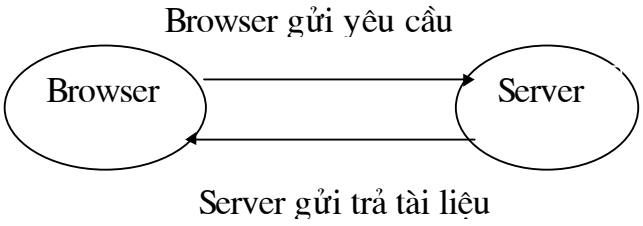
Như chúng ta đã biết sức mạnh của HTML là có khả năng sắp xếp một số lượng thông tin rất lớn các tập tin vào trong cùng một trang. Các tập tin xuất hiện trên một trang về mặt vật lý có thể lưu trú trong cùng một máy tính dưới dạng là chính trang đó, hoặc lưu trú ở một nơi bất kỳ nào khác trên WWW. HTML chỉ chuyên làm nhiệm vụ là tham chiếu vào các tập tin này bằng cách báo cho Browser biết vị trí chính xác của chúng, nên Browser có thể tìm đến chúng một cách nhanh chóng khi cần. Nhưng chúng có nhược điểm là chỉ hạn chế trong phạm vi nội dung tĩnh, nghĩa là những thông tin mà Web hiển thị được chỉ là thông tin không thay đổi ví dụ như các bài báo, đơn thuốc. . . , chúng không thể cung cấp các lệnh máy đặc biệt để máy làm theo, và đặc biệt là chúng không thể khai thác cơ sở dữ liệu bên ngoài vì vậy không đáp ứng được nhu cầu phức tạp của USER. Để đáp ứng được nhu cầu đó người ta đưa ra giải pháp là viết một chương trình có khả năng một mặt giao tiếp với Web Server, mặt khác có thể thao tác được với cơ sở dữ liệu. Một chương trình như vậy có thể gọi là "cổng" (gateway) giữa Web Server và Cơ sở dữ liệu. Chương trình chạy ngoài được cài đặt lên hệ thống mày chủ đó chính là CGI (COMMON GATEWAY INTERFACE).

I Các khái niệm cơ bản

I.1 Tài liệu tĩnh (Static Doccuments)

Đó là kiểu tài liệu được phân phát rất đơn giản từ hệ thống file của Server. Sau đó Phần mềm Web Server sẽ tiến hành tìm kiếm và xác định đúng vị trí file đó trên ổ cứng, mở nó một cách trực tiếp và trả lại kết quả cho Client. Tài liệu tĩnh sẽ là tốt nhất để sử dụng khi thông tin có sẵn trên ổ đĩa cứng, và không thay đổi. Khi cơ sở dữ liệu là nhỏ, cách tiếp cận này có hiệu quả rõ ràng, Server có thể đáp ứng nhu cầu của Client một cách nhanh chóng.

Tuy nhiên nó có hạn chế là không năng động, không đáp ứng nhu cầu thông tin vì vậy không đáp ứng được những yêu cầu phức tạp của người sử dụng. Quá trình phân phát tài liệu tĩnh được thể hiện ở hình 1.1.



Hình 1.1 Phân phát một tài liệu tĩnh

I.2 Tài liệu động (Dynamic doccuments - Doccument on the fly)

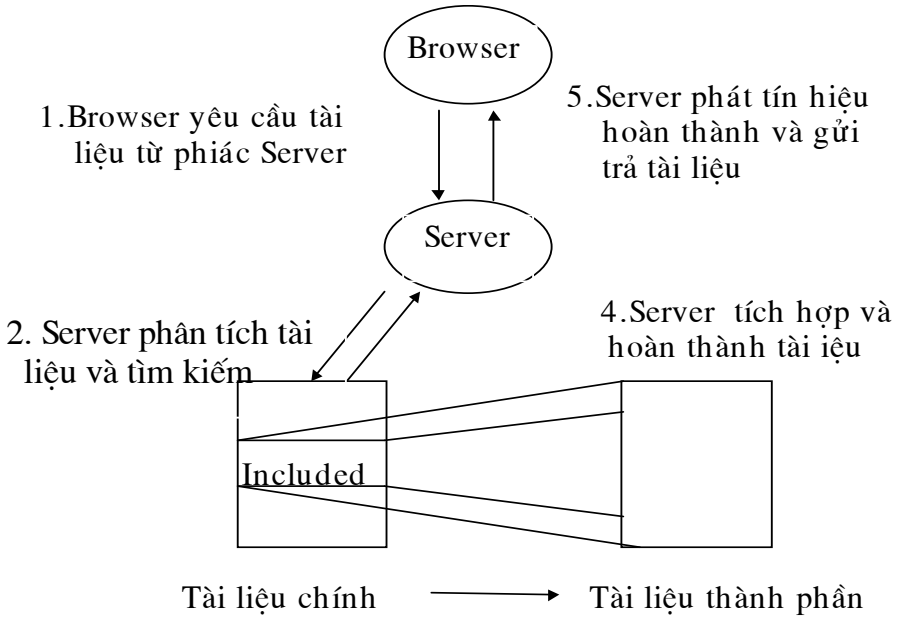
Không giống như tài liệu tĩnh, tài liệu động được sinh ra trong quá trình đang thực hiện "on the fly". Trong trường hợp tài liệu tĩnh, có thể đọc dữ liệu từ một file đang tồn tại thì nói chung tài liệu động có thể không cần quan tâm đến điều đó. Ví dụ, tài liệu động có thể được sinh ra từ cơ sở dữ liệu, từ các phương tiện khoa học nào đó như hệ thống kiểm tra âm lượng. . . Các tài liệu mà thường xuyên thay đổi và gửi tín hiệu một cách trực tiếp đến client như chúng đã được tạo, và sẽ được lưu trữ trong hệ thống file. Trong trường hợp khác, chúng còn có thể trùng khớp với nội dung đã được hoà trộn, với một số lượng nội dung đã được sinh ra khi trang được phân phát một cách thật sự.

Sự khác nhau cơ bản giữa tài liệu tĩnh và tài liệu động đó là tài liệu tĩnh thì được phân phát từ hệ thống file trên đĩa cứng còn tài liệu động thì được sinh ra một cách tạm thời ngay trong thời gian làm việc "on the fly".

I.3 Một cách tiếp cận tới tài liệu động: Công nghệ Server-side include

Hệ thống HTML có thể dễ dàng tạo ra những liên kết bất kỳ với tài liệu nào đó. Tuy nhiên thi thoảng chúng cũng mong muốn có được một tài liệu

HTML lớn được tập hợp từ những tài liệu nhỏ hơn. Đặt ra vấn đề là tại sao một tài liệu Web lại không thể đơn giản chỉ là gồm những tài liệu được tham chiếu đến một bản vật lý chứa đựng bản copy thứ hai. Hàng loạt những version của HTML không cho phép điều này. Tuy nhiên không có gì cản trở được Web Server thực hiện được điều đó miễn là các version của HTML bao gồm các thành phần đã được cho phép. Khi Browser có yêu cầu tài liệu đối với Server, Server phân tích tài liệu và nhìn một cách trực tiếp vào tài liệu chính (main doccument), sau đó Server sẽ truy cập đến tập tài liệu (include document) và lắp ráp tài liệu hoàn chỉnh rồi phát tín hiệu hoàn thành nhiệm vụ và gửi kết quả tới Browser. Cách tiếp cận này gọi là Server site include được thể hiện bằng sơ đồ sau (hình 1.2).



Hình 1.2 Công nghệ Server-side Include

II CGI (Common Gateway Interface)

II.1 CGI là gì:

CGI là một chuẩn dùng để phát triển các ứng dụng động lên trang Web và sử dụng giao thức truyền siêu văn bản (HTTP) đưa ra các nội dung động này tới Browser. Và chuẩn này đã hỗ trợ Web Server thao tác với cơ sở dữ liệu. Tuy nhiên, CGI rất khó sử dụng và khả năng phân phối tương tác rất bị hạn chế, CGI chủ yếu được dùng để truy nhập thông tin thông qua các Form. Khi người sử dụng nhập thông tin vào từ bàn phím, Web Browser gửi các thông tin đó cho Web Server. Web Server nhận các thông tin đó gọi thực hiện một ngữ trình Gateway tương ứng và chuyển các thông tin này cho Gateway thông qua chuẩn CGI. Khi đó các thông tin từ người sử dụng được chuyển tới Gateway thông qua các biến môi trường hoặc dòng nhập chuẩn. Sau đó Gateway phân tích, xử lý các thông tin đó và thực hiện công việc của mình. Cuối cùng Gateway trả về các thông tin cho Web Server để Web Server chuyển các thông tin này tới người sử dụng hoặc lưu giữ dữ liệu trong cơ sở dữ liệu. Gateway có thể là một ngữ trình Script hay một chương trình được viết bằng C/C++,Perl. .

II.2 Mục tiêu của CGI

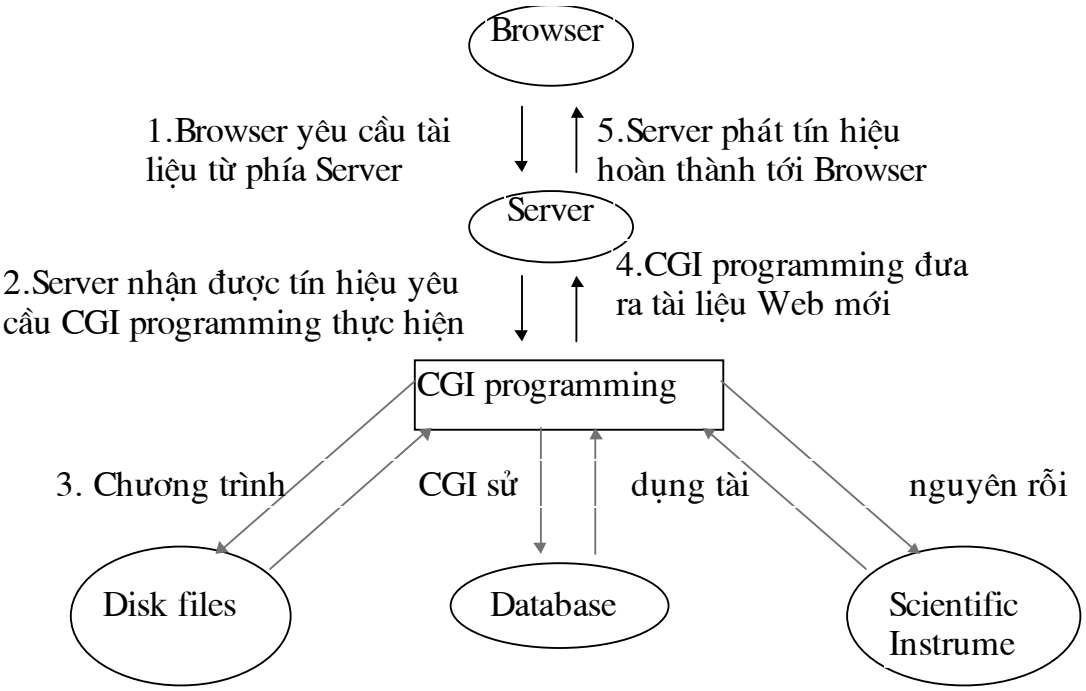
Chuẩn CGI được phát triển bởi NCSA (National Center for Supercomputing Applications) để đáp ứng nhu cầu của người sử dụng bằng cách chạy một chương trình ngoài phù hợp. Trong phần thêm vào một giao diện chuẩn phù hợp, CGI tìm kiếm và suy luận để có thể đảm bảo cho thao tác nhập vào của user, thông thường Form giao diện sẽ không mất vì nhờ giới hạn của hệ điều hành Server. Chuẩn CGI cố gắng cung cấp một chương trình chạy ngoài với thông tin có khả năng về Server và Browser. Trong phần thêm vào ta có thể biết thêm thông tin của user. Chuẩn CGI cố gắng phát triển một ứng dụng CGI thật đơn giản và dễ sử dụng. Phần lớn Standard CGI đều đạt

được những thành công .Rõ ràng cách tiếp cận này là hiệu quả, đặc biệt là khi kết hợp tốt với hệ điều hành, tính dễ dàng thích nghi và đơn giản của CGI Standard làm nó trở nên phổ biến để sử dụng. Nhưng đôi khi nó cũng có những hạn chế nhất định đó là khi dữ liệu đưa vào quá phức tạp, đặc biệt là khi công việc đòi hỏi chính xác và tỉ mỉ. Tuy nhiên nó vẫn rất phổ biến và phát triển với nhiều công cụ có sẵn, đặc biệt là tốt đối với **C& PERN** (CGI programming in C & PERN).

II.3 Cách thức hoạt động của một chương trình CGI

Phần lớn tài liệu động đều tuân thủ theo luật phối hợp của Server. Điều đó có nghĩa là tài liệu động được sinh ra một cách toàn vẹn bởi một chương trình ngoài được thực hiện bởi yêu cầu của User. Chương trình ngoài sẽ tiếp nhận tham số từ văn bản như một đầu vào chuẩn và đưa ra một kết quả như một đầu ra chuẩn. Khi Browser yêu cầu tài liệu từ phía Server, Server nhận được tín hiệu yêu cầu và thực hiện CGI, Chương trình CGI sẽ sử dụng những tài nguyên bên ngoài như đĩa files, cơ sở dữ liệu và các phương tiện khoa học. Sau khi thực hiện xong chương trình CGI đưa ra một tài liệu Web mới. Server phát tín hiệu hoàn thành nhiệm vụ và gửi trả tài liệu cho Browser (Minh hoạ

hình1.3).



Hình 1.3 Cơ chế CGI

III Chuẩn CGI

CGI Standanrd có mục đích xác định sẵn một giao diện giữa Web Server và chương trình. Những điều cần cho một chuẩn đó là tài liệu động bản thân nó tự sinh ra trang Web khi chạy một chương trình. Khi Server thực hiện một chương trình để thoả mãn yêu cầu từ Browser. Browser sau khi đưa ra yêu cầu sử dụng HTTP. Server sẽ trả lời bằng cách hoặc cấp phát một tài liệu, hoặc mã trạng thái, hoặc đưa ra một URL khác được chấp nhận từ kết quả của một giao thức, Vì vậy chương trình CGI thường xuyên cần đến HTTP một cách trực tiếp. Đây là một khía cạnh chính yếu và quan trọng trong CGI

programming. Yêu cầu của HTTP có thể có vài kiểu khác nhau, người ta gọi là phương pháp. Có hai phương pháp chính đó là phương pháp POST và GET.

III.1 Phương pháp GET

Phương pháp được sử dụng khi có một yêu cầu một tài liệu của người sử dụng. Nếu đã yêu cầu một URL cho chương trình CGI thì chương trình CGI sẽ sinh ra một tài liệu mới, một mã lỗi. Chương trình CGI có thể đánh dấu những tình huống đã thực hiện vào biến môi trường REQUEST_METHOD chứa đựng vào xâu GET. Thông tin yêu cầu của người dùng sẽ được lưu trữ trong biến môi trường QUERY_STRING.

III.2 Phương pháp POST

Phương pháp POST được sử dụng để truyền thông tin từ Browser gửi đến Server. Trong phần lớn các trường hợp thông tin yêu cầu được lưu vào biến trong Standard Input. Chương trình CGI sẽ đọc các thông tin trình diện từ Standard input và thực hiện chương trình. Trong trường hợp này biến môi trường REQUEST-METHOD sẽ được đặt vào xâu POST. Sau khi thực hiện nhiệm vụ chương trình được gọi sẽ sinh ra một tài liệu mới, một mã lỗi hay một URL khác.

III.3 Sự khác nhau giữa phương pháp GET & POST

Sự khác biệt cơ bản giữa hai phương pháp này là ở cách truyền dữ liệu dạng Form tới chương trình CGI. Nếu sử dụng phương pháp GET, thì khi Client yêu cầu tới Server xâu QUERY sẽ được ghi tiếp vào URL của chương trình. Ưu điểm của phương pháp này là ở chỗ có thể truy nhập chương trình mà không cần Form. Còn với phương pháp POST thì độ daig dữ liệu sẽ không bị hạn chế như dùng phương pháp GET

III.4 Dòng vào chuẩn (Standard Input)

Một chương trình chạy ngoài chuẩn có dạng dữ liệu sẽ được lưu trữ trong biến môi trường hay thông qua một dòng lệnh. Cách tiếp cận như vậy có thể gặp rủi ro với một hệ điều hành là khi số lượng thông tin quá lớn. Tuy nhiên chuẩn CGI chấp nhận cách tiếp cận này, nó cho phép và cổ vũ cách tiếp cận này với một kiểu dữ liệu thông qua chương trình chạy ngoài như Standard input, có nghĩa là dữ liệu có thể truy nhập thông qua Standard I/O. Trong ngôn ngữ lập trình C gọi là các hàm. Khi không có dữ liệu được trình diện từ người sử dụng hay một form dữ liệu đã được trình diện với phương pháp GET, thì chuẩn vào cũng không chứa đựng thông tin.

Tuy nhiên khi dữ liệu được gửi vào theo phương pháp POST, thì dữ liệu sẽ được xuất hiện trong dòng chuẩn vào (Standart Input)

III.5 Dòng ra chuẩn (CGI Standard Output)

Khi chạy một chương trình CGI thường mong đợi đưa ra kết quả là một trong ba đối tượng như sau:

- Một tài liệu Web đúng đắn: trong trường hợp này cần quan tâm đến kiểu dữ liệu sẽ đưa ra.
- Một mã lỗi: Nếu một lỗi xuất hiện chương trình CGI có thể gửi ra một mã trạng thái của tài liệu, hoặc một thông báo lỗi cho người dùng
- Đưa ra một URL khác: Nếu Server không trực tiếp giải quyết được yêu cầu của người sử dụng thì nó sẽ cung cấp một địa chỉ URL khác.

CHƯƠNG II XÂY DỰNG MỘT CHƯƠNG TRÌNH CGI TRÊN C

I Truyền số liệu cho CGI gateway

Web Server có thể chuyển thông tin cho gateway bằng tham số dòng lệnh, bằng biến môi trường hoặc bằng dòng nhập chuẩn.

I.1 Truyền thông tin qua tham số dòng lệnh

Ta xem xét trường hợp Web Server truyền thông tin cho các gateway qua tham số dòng lệnh (command line argument). Trong trường hợp này, Web Server tách chuỗi tham số dòng lệnh thành các từ riêng rẽ và phân cách chúng bằng các dấu cộng ("+") rồi đặt chúng vào tham số dòng lệnh. Từ đầu tiên của chuỗi yêu cầu sẽ trở thành phần tử đầu tiên ngay sau tên của ngữ trình. Chú ý là nếu chuỗi yêu cầu dài quá độ dài quy định của tham số dòng lệnh thì Server sẽ không ghi giá trị gì vào tham số dòng lệnh mà biến môi trường QUERY_STRING sẽ chứa giá trị đó.

I.2 Truyền thông tin qua biến môi trường

Với trường hợp này, các thông tin về yêu cầu của Web Browser được Web Server truyền cho ngữ trình CGI thông qua các biến môi trường của Server. Phương thức truy nhập các biến môi trường của ngữ trình CGI phụ thuộc vào ngôn ngữ viết nên ngữ trình đó. Nếu một biến môi trường không thích hợp trong ngữ cảnh yêu cầu thì nó sẽ không được thiết lập hoặc sẽ được đặt giá trị là một chuỗi rỗng. Các biến môi trường sau đây được dùng để chuyển thông tin cho Web Server tới các ngữ trình CGI:

- QUERY_STRING: Nếu URL có chứa chuỗi yêu cầu, biến này sẽ chứa giá trị của chuỗi yêu cầu đó.

- CONTENT_TYPE: Biến này sẽ được xác định trong trường hợp nếu dữ liệu được gắn vào yêu cầu và chuyển qua dòng nhập chuẩn. Nó chỉ ra kiểu MINE của dữ liệu đó.
- CONTENT_LENGTH: Chứa giá trị độ dài của dữ liệu nếu dữ liệu được gắn vào yêu cầu và chuyển qua dòng nhập chuẩn của GateWay.
- PATH_INFO: Chứa bất kỳ dữ liệu nào được thêm vào URL
- PATH_TRANSLATED: Chứa thông tin được đưa ra trong biến PATH_INFO nhưng được thêm vào đầu đường dẫn tới gốc của Web Server.
- GATEWAY_INTERFACE: Xác định số hiệu phiên bản của CGI mà Web Server đang sử dụng, dưới dạng tên/số hiệu.
- REMOTE_USER: tên của người sử dụng của máy gửi yêu cầu.
- REMOTE_ADDR: Địa chỉ Internet của máy gửi yêu cầu .
- REMOTE_HOST: Tên của máy gửi yêu cầu
- AUTH_TYPE: phương thức xác thực được Server sử dụng .
- REQUERY_METHOD: Chỉ ra phương thức yêu cầu. Với các yêu cầu HTTP, các phương thức yêu cầu có thể là GET, POST, PUT và HEAD.
- SCRIPT_NAME: Chứa đường dẫn ảo tới ngữ trình đang được thi hành.
- SERVER_NAME: Tên hoặc địa chỉ IP của Web Server.
- SERVER_PORT: Số hiệu của cổng nhận được yêu cầu .
- SERVER_PROTOCOL: Tên và số hiệu phiên bản của giao thức yêu cầu
- SERVER_SOFTWARE: Xác định phần mềm Server đang dùng.

Những thông tin trong phần đầu của HTTP cũng có thể được Web Server chuyển cho ngữ trình qua những biến có tên bắt đầu bằng HTTP. Một số biến thông thường là:

- HTTP_ACCEPT chỉ ra các kiểu MINE, mà Web Browser chấp nhận được

- HTTP_USER_AGENT Chuỗi nhận dạng khách hàng. Thông thường là tên và số hiệu phiên bản của Web Browser.

Hầu hết việc truy cập dữ liệu vào của một chương trình CGI là thông qua các biến môi trường.

I.3 Truyền thông tin qua dòng nhập chuẩn

Nếu một yêu cầu được tạo bởi phương thức HTTP POST, dữ liệu từ Web Browser được Web Server gửi cho ngữ trình CGI (gateway) qua dòng nhập chuẩn của nó. Các kiểu MINE của dữ liệu và độ dài của dữ liệu được chứa trong các biến môi trường CONTENT_TYPE và CONTENT_LENGTH.

II Xử lý các FORM

Xử lý các Form là một trong những ứng dụng quan trọng nhất của CGI. Form do HTML tạo ra cho phép người sử dụng nhập các thông tin hay dữ liệu. Sau khi nhập các thông tin hay dữ liệu đó được gửi tới Server nhằm thức hiện chương trình (có liên quan đến form) để giải mã form đó. Chương trình xử lý thông tin và sau đó gửi trả lại cho người sử dụng.

II.1 Truy cập dữ liệu từ Form

II.1.1 Các xâu query

Một cách để gửi dữ liệu dạng Form tới chương trình CGI là ghi tiếp các thông tin về form vào địa chỉ URL đạt sau dấu hỏi. Các dạng URL có thể như sau:

http://acernt/cgi/name.c?fortune.

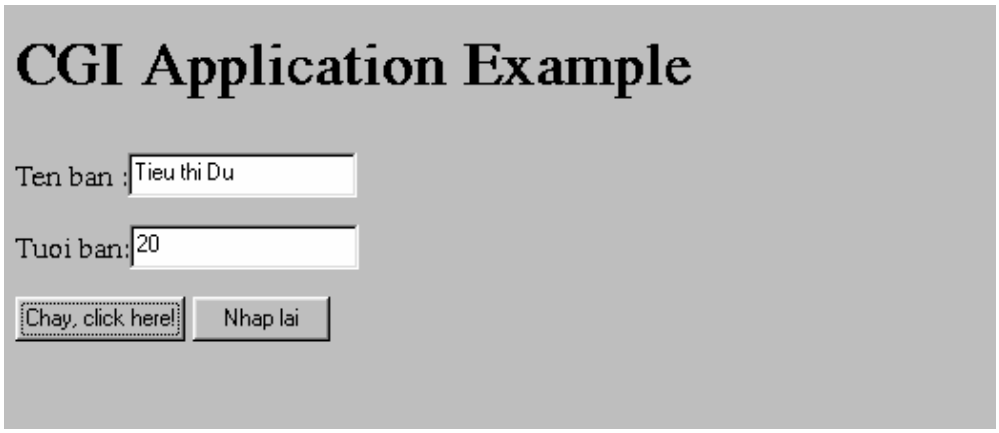
Sau dấu chấm hỏi được gọi là xâu query (query string). Khi chuyển địa chỉ URL và xâu query tới Server, Server sẽ gọi chương trình CGI được chỉ định ở phần URL trước dấu hỏi và lưu trữ ở phần sau dấu hỏi vào biến môi trường

II.1.2 Chương trình xử lý Form

Để thực hiện một chương trình CGI cần phải bắt đầu từ một trang HTML có chứa một URL chỉ đến ứng dụng CGI đó. Một trang HTML đó có thể viết như sau:

```
<html>
<head>
<title>chào bạn</title>
</head>
<body>
<h1>CGI Application Example </h1>
<br>
<form action="http://sco5:7000/cgi/ktra" method="POST" >
Ten ban :<input Name="name_file" type="text"><p>
Tuoi ban:<input name="tuoi" type="text"><p>
<input type="submit" value="chay,click here!">
</form>
</body>
</html>
```

Form nhập dữ liệu:



Trong Form trên ta thấy có hai nút: ***Chạy, Click here*** và ***Nhập lại***

Nút ***Chạy, Click here*** dùng để chuyển những thông tin trong Form tới chương trình CGI. Sau khi nhập những thông tin cần thiết và chọn ***Chạy, click here*** ta sẽ nhận được kết quả về những thông tin trạng thái như độ dài chuỗi yêu cầu, phương pháp truy nhập, tên máy chủ, giao thức sử dụng . . .do chương trình CGI cung cấp.

Nút ***Nhập lại*** dùng để xoá các thông tin đã điền trong Form.

Nội dung chương trình CGI (ktra.c) được viết bằng ngôn ngữ C trình bày chi tiết trong phần phụ lục.

Kết quả trả lại của chương trìng CGI trên màn hình Web Browser:

TEST TRANG THAI

Content_Length	23
Content_Type	application/x-www-form-urlencoded
Gateway_Interface	CGI/1.1
http_accept	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
http_referer	file:C:\DUTT\LV\kt.htm
path_info	
query_string	
remote_addr	190.2.3.2
request_method	POST
script_name	/cgi/ktra
server_name	sco5
server_port	7000
server_protocol	HTTP/1.0
server_software	Oracle_Web_listener/1.01fc5

InputBuffer: Ten=Tieu+thi+Du&Tuoi=20

Variables

Tuoi	20
Ten	Tieu thi Du

II.2 Hoạt động của chương trình CGI

II.2.1 Lấy dữ liệu từ Form và xử lý dữ liệu

Sau khi người dùng nhập dữ liệu vào Form và trình diện lên Server, nếu sử dụng phương pháp yêu cầu là phương pháp POST thì Web Server sẽ xác định một số giá trị tương ứng vào một số biến môi trường và đưa dữ liệu của người dùng vào trong dòng vào chuẩn (Standard Input) của chương trình CGI. Khi đó chương trình CGI tham khảo các biến môi trường rồi lấy dữ liệu từ dòng vào chuẩn đó để giải quyết yêu cầu. Còn nếu trình diện yêu cầu bằng phương pháp GET thì ngoài việc đặt giá trị cho các biến môi trường thông thường, Web Server đưa dữ liệu nhận được từ người dùng vào biến môi trường QUERY_STRING, chương trình CGI lấy dữ liệu từ đó.

Cụ thể với chương trình **ktra.c** là chương trình kiểm tra trạng thái Web Server, khi người dùng nhập các thông tin tên, tuổi và trình diện lên Web Server, và chương trình sẽ nhận được dữ liệu từ Web Server. Chương trình gồm có các thủ tục sau:

- Thủ tục strcvrt có chức năng chuyển đổi kí tự thành dạng xâu.
- Thủ tục TwoHex2Int chuyển đổi mã ESCAPE thành kí tự.
- Thủ tục urlDecode giải mã dữ liệu
- Thủ tục Main đọc dữ liệu từ Stdin và đưa ra dữ liệu dưới dạng HTML chuẩn.

Trước tiên chương trình CGI sẽ tiến hành kiểm tra xem phương thức yêu cầu của Client là phương thức nào bằng cách đọc dữ liệu trong biến môi trường REQUES_METHOD với dòng lệnh:

```
pRequestMethod = getenv("REQUEST_METHOD") ;
if (pRequestMethod == NULL || pRequestMethod[0] == '\0')
{
     printf("\nERROR:Request Method error\n") ;
     goto error ;
}
if ( strcmp( pRequestMethod, "POST" ) == 0 )
.....
```

Ngoài việc đọc biến môi trường REQUEST_METHOD, chương trình CGI còn có thể tham khảo một số biến môi trường khác nếu nó thấy cần. Ví dụ muốn biết thông tin về phần mềm Server đang sử dụng thì ta đọc dữ liệu từ biến môi trường SERVER_SOFWARE bằng dòng lệnh:

```
p = getenv("SERVER_SOFTWARE") ;
if ( p != NULL && *p != '\0' )
     printf(p) ;
else
     printf(" ") ;
```

Do dữ liệu được gửi lên theo phương theo phương thức POST nên chương trình CGI sẽ đọc biến môi trường CONTENT_LENGTH để biết độ dài dữ liệu rồi tiến hành đọc dữ liệu từ Standard Input và xử lý dữ liệu.

```
p = getenv("CONTENT_LENGTH") ;
if ( p != NULL && *p != '\0' )
     ContentLengh = atoi(p) ;
else
     ContentLength = 0 ;

i = 0 ;
while ( i < ContentLength )
{
     x = fgetc(stdin) ;
     if ( x == EOF )
          break ;
     InputBuffer[i++] = x ;
}
```

Sau nhận được dữ liệu chương trình CGI sẽ tiến hành giải mã dữ liệu đó (vì một số ký hiệu đặc biệt đã được mã hoá) bằng thủ tục *urlDecode*, thủ tục đó được viết như sau:

```
void urlDecode( char *p )
{
     char    *pD = p ;

     while (*p)
     {
          if ( *p == '%' )
          {
               p++ ;
               if ( isxdigit(p[0]) && isxdigit(p[1]) )
               {
                    *pD++ = (char) TwoHex2Int(p) ;
                    p += 2 ;
               }
          }
          else
          {
               *pD++ = *p++ ;
          }
     }
     *pD = '\0' ;
}
```

II.2.2 Đưa kết quả đưa ra từ CGI Gateway

Kết quả trả về từ ngữ trình CGI (gateway) được Server nhận và chuyển nó cho người gửi yêu cầu (Web Server). Khi người sử dụng gọi URL của một

chương trình CGI nào đó và gửi tới Server để tìm file, nếu Server nhận ra địa chỉ được yêu cầu là một chương trình CGI, Server sẽ không trả lại toàn bộ nội dung file mà thay vào đó nó sẽ chạy chương trình.

Các Gateway CGI muốn tạo ra các tư liệu thông tin để trả về cho người sử dụng phải thông báo cho Web Server về loại thông tin mà nó gửi cho Server dạng như sau:

Content_type: type/subtype

Type và Subtype là kiểu MINE cho thông tin mà Gateway cần gửi. Nếu cần gửi một tệp dạng văn bản (ASCII) thì type/submit phải là "text/plain ", còn nếu cần gửi một tư liệu HTML thì type/subtype phải là "text/html".

Trong chương trình **ktra.c** do muốn lấy kết quả trả về dưới dạng HTML chuẩn nên gửi thông báo cho Web Server biết dạng thông tin cần trả về cho người dùng bằng cách:

printf("Content-Type: text/html\n\n") ;

Theo dòng lệnh này, Web Server tự động trả lại kết quả thực hiện chương trình **ktra.exe** dưới dạng HTML chuẩn (xem hàm main() của chương trình ktra.c ở phụ lục 1).

Các Gateway cũng không nhất thiết phải trả về một tư liệu mà nó có thể trả về một URL tới một tập tin hay một thông tin khác. Khi đó Web Server sẽ dựa vào URL này để xác định và lấy thông tin hay tệp đó rồi gửi nó cho Web Browser. Để thực hiện được công việc này các gateway phải gửi cho Server dòng sau đây:

Location: URL address

II.2.3 Thông tin kết quả từ chương trình CGI:

Như ta đã biết ở phần trên, khi chạy một chương trình CGI, Server thay vì đưa ra văn bản tĩnh sẽ đưa ra kết quả của chương trình. Tuy nhiên, vấn đề

là ở chỗ chương trình CGI phải làm sao tạo thông tin ra để phù hợp nhất với Browser.

Thông tin thông thường nhất do một chương trình CGI tạo ra là một văn bản đơn giản ở dạng plain text hay HTML để cho Browser hiển thị như đối với các văn bản khác trên Web. Tuy nhiên CGI còn có khả năng cung cấp các tiện ích như sau:

- Trả lại đồ hoạ hay các dữ liệu nhị phân khác
- Chỉ cho Browser biết có cất văn bản hay không
- Gửi các mã trạng thái HTTP đặc biệt tới Browser
- Chỉ cho Server gửi một văn bản có sẵn

Các kỹ thuật trên đòi hỏi cần phải biết thêm đôi chút về các Header của chương trình CGI.

II.2.4 Các Header CGI

Header thường thấy nhất là Content-type, là HTTP header chứa kiểu nội dung MINE mô tả dữ liệu. Ngoài ra các header khác còn có thể mô tả:

- Kích thước dữ liệu
- Các văn bản khác mà Server phải trả về
- Các mã trạng thái HTTP

Sau đây là một số header thông thường:

1. Header Content-length: mô tả độ dài (theo bytes) luồng dữ liệu ra. Sử dụng dữ liệu nhị phân.
2. Header Expires: Mô tả ngày giờ của của các văn bản không còn giá trị và Browser cần tải lại (reload)
3. Header Location: Định hướng lại cho Server.
4. Header Pragma: Chỉ định có cất văn bản đi hay không
5. Header Status: Trạng thái của yêu cầu.

Dưới đây sẽ tìm hiểu sâu một chút về các header trên:

+ Accept types và Content Types

Các chương trình CGI có thể trả lại gần như bất cứ dạng văn bản nào mà Client có thể xử lý được: ví dụ như file text, file HTML hay có thể cả Postscript, PDF, SGML ..v.v. .Do vậy Client sẽ chuyển danh sách các kiểu file nó chấp nhận tới Server khi Server yêu cầu. Server sẽ lưu trữ thông tin này vào biến môi trường HTTP_ACCEPT và chương trình CGI có thể kiểm tra biến này để bảo đảm rằng đã trả về một file có dạng mà Browser có thể xử lý được. Khi trả về một tài liệu, chương trình CGI cũng cần phải sử dụng Header Content- type để chỉ cho Client biết nó đang gửi loại dữ liệu nào, như vậy Browser có thể định dạng và hiển thị văn bản một cách chuẩn xác.

+ Header content-length

Header này chỉ định kích thước dữ liệu định truyền đi. Sử dụng header này cho phép tránh được các lỗi dữ liệu từ Server khi đang xử lý dữ liệu nhị phân bởi Server sẽ biết được số byte từ luồng dữ liệu.

+ Sử dụng Header Location để định hướng lại Server

Chương trình CGI có thể lệnh cho Server lấy một văn bản đã có sẵn và hiển thị văn bản đó quá trình này gọi là định hướng lại Server.

Lý do để người ta sử dụng kĩ thuật này là nhằm trả lại một văn bản tính sau khi người sử dụng đã thực hiện một thao tác nào đó. Ví dụ như sau khi họ đã điền vào một Form bạn muốn hiển thị một vài dòng cám ơn. Về nguyên tắc chương trình CGI sau mỗi lần gọi có thể tạo và hiển thị Message đó, nhưng hiệu quả hơn vẫn là gửi các câu lệnh cho Server để định hướng lại và lấy một file có chứa Message cám ơn đó.

Để định hướng lại Server, người ta sử dụng Header Location để chỉ cho Server biết cần phải chuyển những gì và Server sẽ lấy văn bản đó.

+ Các header Expires và Pragma

Hầu hết các Browser sẽ cất trong cache văn bản mà truy nhập nhằm tiết kiệm tài nguyên vì mỗi lần tìm văn bản đó Browser sẽ không lấy lại văn bản đó nữa.

Tuy vậy, đối với các văn bản ảo do chương trình CGI tạo ra thì đây sẽ là một điều phiền phức vì khi Browser truy nhập vào một chương trình do CGI tạo ra thông thường nó sẽ cất văn bản đó. Các lần sau khi truy cập văn bản đó thì Client sẽ không yêu cầu Server mà sử dụng luôn văn bản đã cất do vậy thông tin cung cấp cho người sử dụng có thể không chính xác nữa ví dụ ngày tháng- có trong văn bản lại là của lần truy nhập trước và do đó sẽ không còn giá trị. Để hạn chế nhược điểm đó người ta sử dụng các Header Exprires và Pragma để cho Client không cất văn bản đó đi.

+ Các mã trạng thái

Giao thức HTTP sử dụng các mã trạng thái để liên lạc với trạng thái của các yêu cầu chẳng hạn nếu văn bản cần truy cập không tồn tại thì sẽ trả về mã trạng thái "404" tới Browser và nếu văn bản đã bị rời đi nơi khác thì trả về mã "301".

Header Status chỉ các mã trạng thái gồm có 3 số, tiếp theo là xâu chỉ nọi dung mã trạng thái đó, ví dụ:

- Mã 200: Truy cập thành công
- Mã 204: Không có tín hiệu trả lời
- Mã 301: Văn bản đã bị chuyển
- Mã 401: Không được quyền truy cập
- Mã 404: Không tìm thấy
- Mã 500: Lỗi bên trong Server
- Mã 501: Không sử dụng (not implemented).

CHƯƠNG III ORACLE WEBSERVER VÀ XÂY DỰNG CHƯƠNG TRÌNH CGI TRUY NHẬP CSDL ORACLE

A. ORACLE WEBSERVER

I Kiến trúc của Oracle Web Server

Oracle WebServer bao gồm các thành phần chính như sau (thể hiện trong hình vẽ (2.1) :

+ Web Listener

+ Web Request Broker

+ Sercure Socket Layer

+ Web Server Manager

+ CGI Interface

+ PL/SQL Agent

I.1 Web Listener

Là một giao thức mạnh trong việc giải quyết yêu cầu và đưa ra tài liệu siêu phương tiện gửi tới Web Browser. Nó hỗ trợ tất cả những chuẩn chức năng Web Server như:

+ Hệ thống file ảo
+ Domain Name Services
+ Hỗ trợ về Sơ đồ ảnh
+ Hỗ trợ về CGI
+ Giao thức kết nối an toàn
+ Bảo vệ file và phân quyền cho user

I.2 Web Request Broker:

Đây là phần trọng tâm của WebServer bao gồm :

+ WRB Dispatcher

+ WRB Cartrigger

+ WRB Services

Một câu hỏi đồng bộ cho một ứng dụng của Oracle WebServer sử dụng để thực hiện những ứng dụng trên Server.

I.3 Sercure Sockets Layer

SSL là chuẩn cho an toàn dữ liệu trên mạng. Một vấn đề có ảnh hưởng đến việc kết giao thông tin trên mạng đó là mọi kết nối giữa hai máy tính trên mạng giải quyết nhiều bước trung gian với hàng loạt máy tính từ khi tiếp nhận và quay trở lại thông tin một cách thành công cho đến khi tìm được đến đích. Tiến trình này được gọi là *routing* là cơ sở chủ yếu cho toàn bộ việc kết nối mạng, và bất kỳ máy tính nào trong "chuỗi dẫn đường" hoàn thành việc truy cập dữ liệu.

I.4 Quản trị Web Server

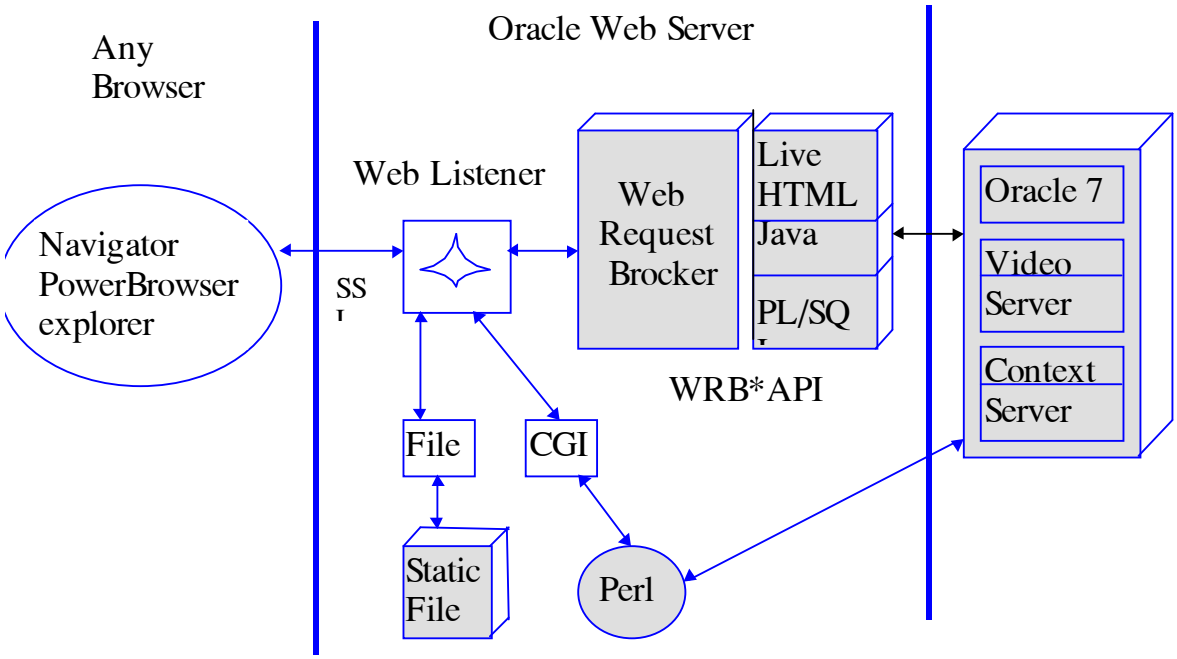
Để giúp đỡ việc quản lý Web site, Oracle Web Server cung cấp một tập hợp trang Web có thể sử dụng chúng để tiến hành làm nhiệm vụ quản lý một cách có hiệu quả nhất. Có những trang đơn giản là soạn thảo những files cấu hình Web Server sử dụng, bạn có thể sử dụng công cụ khác để soạn thảo files một cách trực tiếp.

I.5 Giao diện CGI

Một công nghệ chuẩn được sử dụng bởi Web Listener thực hiện một chương trình ngoài như (C, PERL) sinh ra HTML Doccument.

I.6 PL/SQL Agent

Đây là chương trình Oracle WebServer sử dụng để thực hiện những thủ tục được viết trên PL/SQL, ứng dụng của Oracle trên Oracle7 Server.



Hình 2.1 Kiến trúc Oracle Web Server

II Nguyên tắc hoạt động của Oracle Web Server

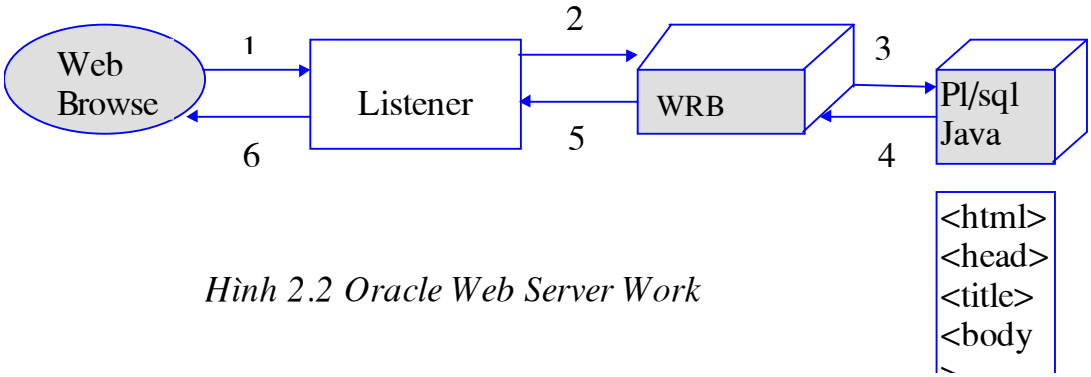
Oracle WebServer là một HTTP với một cơ sở dữ liệu không định trước. Khi WebServer tiếp nhận một URL từ Browser trên WWW hoặc từ mạng cục bộ sử dụng giao thức Web, nó chứa đựng thông tin từ cơ sở dữ liệu hay file hệ thống cần thiết để trả lời yêu cầu. Hệ thống file sử dụng Static Web pages, hay cho CGI Script. Cơ sở dữ liệu sử dụng cho trang Web sinh ra một "live" data. Server Web Listener chấp nhận một URL từ Web Browser và gửi ra ngoài khi Web Listener tiếp nhận URL. Web listener xác định câu hỏi và sử dụng những dịch vụ truy cập thông qua WRB (Web Request Brocker)

một chương trình được truy cập bằng CGI Interface hay truy cập thông qua file hệ thống của công nghệ trên Listener.

Khi Web Browser gửi yêu cầu dưới dạng URL tới Web Listener, Web Listener sẽ tiếp nhận phân tích URL và xác định dịch vụ thực hiện yêu cầu hoặc thông qua WRB. Nếu yêu cầu một tài liệu tĩnh thì tài liệu đó sẽ được lấy từ hệ thống files. Nếu yêu cầu là giành cho một ứng dụng của CGI thì tiến trình CGI sẽ hoạt động. Nếu Web Listener không đáp ứng được yêu cầu thì sẽ gửi qua WRB sau đó WRB sẽ gửi yêu cầu đó tới Cartridger như PL/SQL, Java, LiveHTML. Nếu WRB truy cập ngoài thì Listener sẽ thông qua câu hỏi cho WRB Dispatcher cho một tiến trình, sau đó quay trở lại giải quyết nhiệm vụ .

WRB Dischaper tự thực hiện yêu cầu với sự giúp đỡ của một "xích" (pool) của tiến trình được gọi đó là được gọi là WRB Executable Engines (WRBXs). Một giao diện khác WRBX là quay trở lại sử dụng ứng dụng WRB API. Có những ứng dụng được gọi là WRB cartridges. WRB API đã được thiết kế . Sự kết hợp giữa một Cartridges và WRB API tạo ra một WRB Service. Thông thường thì có 3 loại dịch vụ mà Oracle WebServer hỗ trợ là :

- PL/SQL Cartridges : Thực hiện các thủ tục PL/SQL sinh mã HTML đồng thời sử dụng Oracle Data.
- Java cartridges : Thực hiện Java trên Server.
- Live HTML : Hiện ra Web page. Web page được thực hiện bởi Hệ điều hành.



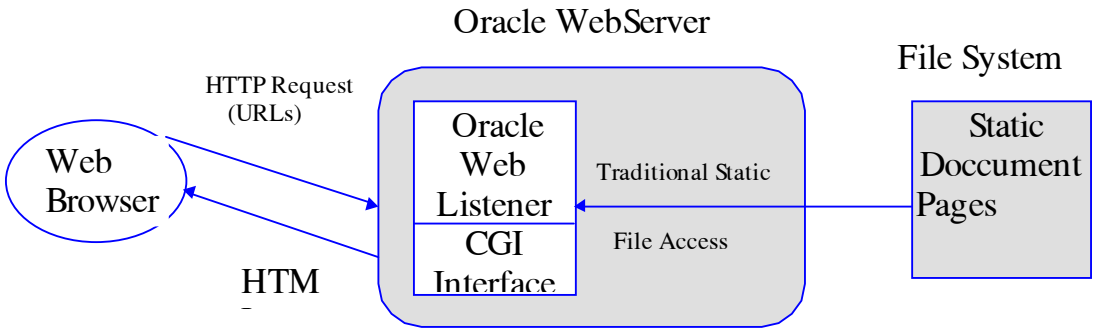
Hình 2.2 Oracle Web Server Work

Giải thích hình 2.2

1. *Web Browser đưa ra yêu cầu URL cho Web Listener*
2. *Web Listener gửi yêu cầu tới WRB*
3. *Nếu PL/SQL được chọn thì sẽ móc nối vào cơ sở dữ liệu*
4. *Thủ tục PL/SQL sẽ sinh tài liệu HTML*
5. *PL/SQL Agent thông qua tài liệu HTML tới Web Listener*
6. *Web Listener gửi tài liệu HTML tới Browser*

II.1 The Web Listener (OWL)

Oracle Web Listener là một giao thức truyền thông HTTP. Web Listener có nhiệm vụ tiếp nhận yêu cầu từ Web Browser gửi đến WRB và sau đó lại nhận kết quả để gửi trả cho Browser. Cho phép xử lý đồng thời nhiều câu hỏi trong cùng thời gian thông qua chuẩn HTTP hay HTTP trên SSL. Web Listener và Web Client kết nối với nhau thông qua giao thức HTTP.(Hình vẽ 2.3)



Hình 2.3 Oracle Web Listener

Mỗi tiến trình Oracle Web Listenner đều chấp nhận kết nối nhiều Web Browser trên một hay nhiều địa chỉ IP/ hoặc cổng kết hợp sử dụng HTTP để giải mã yêu cầu từ siêu văn bản và giao thức điều khiển truyền thông TCP/IP (Transmission Control Protocol) /giao thức Internet (Internet Protocol) được sử dụng như một giao thức kết nối lớp dưới. Một số tiến trình Web Listener có thể chạy trên một máy vào cùng một thời điểm.

Thông thường khi Web Listener mở một file đã được yêu cầu, File sẽ mở và ánh xạ vào bộ nhớ trong cho đến khi Clients sử dụng xong và kết thúc nó. Web Listener sẽ đóng file và giải phóng memory mapping kết nối với nó. Web Listener cho phép xác định rõ các file ở trong **cache.** Cached file sẽ mở khi client yêu cầu chúng.

Về vấn đề an toàn dữ liệu, Oracle Web Listener cho phép tạo file ảo hay thư mục ảo bởi **Authentication Scheme** hay **Restriction Scheme** để bảo vệ chúng.

II.1.1 Authentication Scheme

Khi một file hay một thư mục được bảo vệ bởi **Authentication Scheme**, một Client có nhu cầu truy cập thì phải cung cấp Username và

Password. Vậy thì Authentication Scheme cho phép tạo tên của người dùng (nhóm người dùng) và Password của họ.

Web Listener hỗ trợ hai Authentication Scheme đó là: **Basis Authentication** và **Digest Authentication.** Cả hai lược đồ đều chính xác, **Digest Authentication** thì gửi password từ Client đến Server dưới dạng mã hoá được gọi là **Digest** ngược với **Basis Authentication** thì gửi password không hề mã hoá như vậy độ an toàn sẽ bị giảm đáng kể.

Một vài Web browser không hỗ trợ **Digest Authentication**, nhưng các file và thư mục đòi hỏi Authentication, vì vậy nên sử dụng digest authentication bất cứ khi nào có thể.

II.1.2 Restriction Scheme

Trường hợp một file hay một thư mục được bảo vệ bởi **Restriction** Scheme, thì chỉ Client đang truy cập Web Listener từ một nhóm an toàn mới có thể truy cập nó. Có hai **Restriction Scheme** mà Web Listener hỗ trợ đó là **IP-based restriction** và **Domain-based restriction.**

+ IP-based restriction: Cho phép định nghĩa nhóm an toàn được đánh dấu bởi IP address.

+ Domain-based restriction: Ngược với IP-based restriction, cho phép định nghĩa nhóm an toàn bởi DNS host hay Domain name.

Ngoài vấn đề an toàn Oracle Web listener còn hỗ trợ những dịch vụ khác như chúng có thể duy trì một vài kiểu của tài liệu ở những dạng khác nhau và cung cấp cho Client những dạng mà chúng yêu cầu.

Ví dụ nếu Client yêu cầu tài liệu bằng tiếng Pháp, Web Listener sẽ kiểm tra xem có Version tiếng Pháp không, nếu có sẽ gửi trả kết quả cho Client, ngược lại nếu không có sẽ trả kết quả ngầm định, thông thường là tiếng Anh.

Một Client có thể yêu cầu tài liệu của một kiểu đặc biệt Multipurpose Internet mail Extensions (MINE). Nếu Web Listener có thể tìm thấy một file yêu cầu trong yêu cầu kiểu MINE , nó sẽ trả lại kiểu đó cho Client ngược lại nó sẽ trả lại kiểu của file có cỡ nhỏ nhất.

Hơn nữa Web Listener có thể duy trì kiểu của tài liệu được mã hoá bởi chương trình nén. Ví dụ nếu một client có thể giải nén một tài liệu bị nén bởi một chương trình. Web Listener có thể trả lại cho Client kiểu nén cuả tài liệu thay cho việc giải nén tài liệu đó.

Web Listener sử dụng filename Extention để đánh dấu dạng của file với khả năng miêu tả ngôn ngữ. Kiểu MINE, hay kiểu mã hoá dữ liệu, dễ dàng hơn cho việc bảo vệ và thông báo cho Clients.

Tóm lại việc sử dụng Oracle Web Listener, Web site của ta có thể trả lời câu hỏi của Client bởi tài liệu HTML động. Cũng giống như kỹ nghệ HTTP, Oracle Web Listener cung cấp cho chúng ta một giao diện gọi là Oracle Web Request Brocker (WRB), cho phép Client chạy chạy chương trình trên Server và trả lại dữ liệu hiệu quả hơn CGI cho phép. Web Listener thông qua mục đích của câu hỏi cho chương trình WRB Dispacher, với sự duy trì bảo vệ của một nhóm các tiến trình.

II.2 The Web Request Broker (WRB)

The WRB là câu hỏi không đồng bộ thực hiện với một ứng dụng API (Application Program Interface) tạo một giao diện động và không liền mạch với công nghệ "back-end" không giống nhau được gọi là " WRB Service". WRB bao gồm:

- WRB Dispatcher
- WRB Service

- WRB Cartridges

II.2.1 WRB Dispatcher:

Listener tiếp xúc với Dispatcher, một tiến trình chạy cùng Listener. Khi tiếp nhận URL đòi hỏi WRB để phân phối dịch vụ yêu cầu giữa một vài tiến trình đang chạy như là WRBXs(Web Request Broker Excutable Engines). Kết quả là Listener được giải phóng để tiếp nhận và xác nhận URLs có hiệu lực đang được đưa vào, các câu hỏi khác thực hiện ở phía sau.

II.2.2 WRB Service

WRB hỗ trợ các dịch vụ như sau:

1. Listener tiếp nhận một URL đòi hỏi WRB và phân tích nó đến Dispatcher
2. Dispatcher tìm kiếm một WRBX rỗi để thực hiện các dịch vụ
3. WRBX đưa ra một file một cách nhanh chóng tới Web Listener
4. Listener phục vụ các file cho Web Client sử dụng HTTP.

II.2.3 WRB Cartridges

WRB API đã được thiết kế như thành phần thứ ba có thể viết như chính Cartridge để mở rộng WebServer.

Mỗi khi tiếp nhận URL thì cần gọi WRB, Web Listener thực hiện hiện câu hỏi thông qua **WRB Dispatcher** hay **Dispatcher. Dispatcher** đảm bảo việc kết nối với một nhóm các tiến trình gọi là WRBXs (Web Request Brocker Excutable Engines). Một giao diện khác WRBX đó là thông qua WRB API tới một WRB Cartridge có thể có các kiểu sau:

- The PL/SQL Agent: Thực hiện những câu lệnh PL/SQL chứa trong cơ sở dữ liệu. Nhìn một cách lạc quan thì nó có thể tốt hơn Java Cartridge nhưng lại không có tất cả những chức năng mà Java có.
- The Java Interpreter: Thực hiện Java trên Server sinh ra trang Web động. Có thể thực hiện PL/SQL thiếu Java nếu sử dụng Cartridge này.
- The Live HTML Interpreter: Đây là sự thi hành trên phạm vi rộng của công nghệ Server Side Includes. Với Live HTML có thể tính đến cả trang Web đưa ra ngoài với bất kỳ một chương trình nào của hệ điều hành cũng có thể thực hiện được.

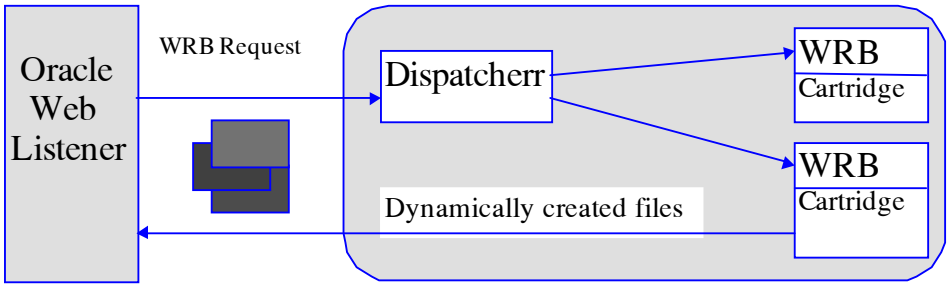
Sự kết hợp giữa Cartridge và WRB API tạo thành một WRB Service. WRBXs là một trường hợp đặc biệt của WRB Services. Có ba kiểu dịch vụ WRB tương ứng với 3 WRB Cartridge trong khi WRBXs đã tạo và phá huỷ nó tuỳ theo công việc.

Dispatcher tạo và bảo vệ WRBXs, dữ liệu thông qua một WRBX bao gồm:

- The URL khởi sự một quá trình
- Ngôn ngữ mong ước của kết quả
- Ký tự mong ước của một kết quả (Ngôn ngữ đã được mã hoá ví dụ như ISO hay Unicode)
- Biến môi trường CGI cho phép cho phép WRB Cartridge chạy một ứng dụng được viết bởi CGI.
- Nếu câu hỏi dính dáng đến việc sử dụng PL/SQL Agent thì DCDs (Database Connection Descriports) được sử dụng.

Dispatcher liên tục điều chỉnh công việc tải số lượng WRBXs đang chạy trong thời gian cho phép, đối tượng khai báo chính. Có những khai báo được đặt bởi người quản trị WebServer, người quyết định số lượng WRBXs chạy là lớn nhất hay nhỏ nhất. Dispatcher tạo một tạo một WRBXs theo yêu cầu và kết nối chúng với dịch vụ WRB phù hợp.

Dispatcher giữ dấu vết của WRBXs đang thực hiện yêu cầu và WRBXs tự do. WRBXs lắp ráp các tiến trình đơn để liên lạc với WRB Dispatcher sử dụng cơ chế dòng dữ liệu phù hợp với hệ điều hành (như là một **PIPE** trong UNIX).



Hình 2.4 Web Request Broker

II.3 rcure Socket Layer (SSL)

Là một chuẩn an toàn trên mạng. Có 3 khía cạnh an toàn đó là:

1. Mã hóa: Một công nghệ phục vụ cho việc mã hoá và giải mã dữ liệu .
2. Tính chính xác: Minh chứng cho sự đúng đắn của thông tin
3. Tính toàn vẹn dữ liệu: Một công nghệ phục vụ cho việc thẩm tra lại toàn bộ dữ liệu đã được chuyển giao và chỉ dữ liệu đã được chuyển giao trọn vẹn và được chấp nhận một cách đúng đắn.

Việc thực hiện một SSL của Oracle Web Server sẽ có vấn đề khi dữ liệu gửi từ Server đến Clients bị mã hoá (Web Browser Programs). Oracle Web Server đã đưa ra một cách khắc phục là Server vẫn gửi dữ liệu dưới dạng bị mã hoá và Clients có thể phục hồi những thông tin bị mã hoá khi tiếp nhận những thông tin đó.

Một hệ thống mã hoá truyền thống được gọi là hệ thống khoá bí mật **(Secret-key)** sử dụng các số được gọi là khóa mã hoá và giải mã thông tin được sử dụng. Hệ thống khoá bí mật này là cực kỳ chắc chắn.
SSL sử dụng một dạng của sự mã hoá gọi là **Public-key encryption** để mã hoá và giải mã dữ liệu, không giống hệ thống mã hoá **seckret-key**, hệ thống Public-key sử dụng cả hai khoá (key pairs). Một khoá được gọi là khoá công cộng **(Public-key)** dùng để mã hoá thông tin, còn khóa kia được gọi là khoá riêng **(Private key)** sử dụng để giải mã thông tin. Cả hai khóa đều là các số tự do liên hệ chính xác với nhau.

Nếu muốn nhận thông tin đã được mã hoá sử dụng khoá cộng cộng thì đầu tiên là phải chạy chương trình sinh ra bởi một trong hai khoá, sau đó phải công bố khoá công cộng trong một cơ sở dữ liệu hay một thư mục, và chứa khoá riêng trong một vùng an toàn trên máy tính. Nhưng nó có một nhược điểm là sự mã hóa của khoá công cộng phụ thuộc hoàn toàn vào độ bí mật của khoá riêng.

Bất kỳ một ai muốn gửi một thông tin đã được mã hoá thì đều phải nhìn vào khoá công cộng trong một thư mục, sử dụng mã hoá thông tin và gửi tới thông tin đã được mã hoá. Chỉ có khoá riêng của bạn mới có thể giải mã được thông tin đó. Vì vậy nếu bạn giữ khoá riêng của bạn một cách bí mật thì không một ai khác có thể đọc được thông tin của bạn.

Bởi vì khoá công cộng để mã hóa thì sẽ chậm hơn khoá bí mật (secret-key). SSL sẽ chỉ được sử dụng nó khi Client đầu tiên kết nối vào WebServer để thay đổi khoá bí mật nó được gọi là một **session key,** với cả Client và Server đều sử dụng sự mã hoá và giải mã dữ liệu.

Một ứng dụng khác của sự mã hoá đó là sự chính xác. Sự chính xác sử dụng mã hoá khoá công cộng bao gồm việc sử dụng **chữ ký điện tử,** công việc này tương tự như chữ ký bằng tay.

Nếu chúng ta muốn "ký" một tài liệu điện tử trong một khuôn khổ ràng buộc về mặt pháp lý, thì cần phải tiến hành với cả hai khoá. Đầu tiên phải chạy một chương trình sinh ra **chữ ký điện tử** có sử dụng khoá riêng và tài liệu của chính nó. Sau đó có thể gắn chặt **chữ ký điện tử** với tài liệu và gửi nó đi. Nếu bất kỳ một ai muốn tiếp nhận tài liệu đó thì cùng với **chữ ký điện tử** có thể sử dụng cả khoá công cộng của bạn để thẩm tra đặc tính của bạn,và tài liệu đó không bị là xáo trộn .

Cuối cùng là việc chứng nhận quyền (Certificates and Certifying Cuthorities) xảy ra khi Clients kết nối vào Website để chuyển giao đòi hỏi của chúng. Trước tiên có thể kiểm tra Username/Passwork nếu đúng WebServer sẽ cấp phát quyền trước khi sự chuyển giao có thể tiến hành.

II.4 Quản lý Web Server

The WebServer có thể sử dụng từ bất kỳ một Web Browser nào, nó có những thành phần chính như sau:

- The Listener Pages
- The WRB Pages
- The PL/SQL Agent Pages
- The Oracle7 Server Pages

II.4.1 Listener Pages

Nhóm giao thiệp rộng nhất với sự quản lý đó là **Web Listener.** Tuy nhiên có những dạng chỉ khi đã nhập đầy giá trị cho những khai báo cấu hình

khác nhau. Trừ những trang an toàn, cần phải định rõ lược đồ an toàn sẽ sử dụng bao gồm các thao tác sau.

- Nếu chọn Restriction thì địa chỉ IP hay Domain names sẽ trao quyền truy cập.
- nếu chọn Authentication thì phải nhập tên và Password thì mới được truy cập.

II.4.2 WRB Pages

Có những trang ta cần phải xác định rõ thư mục thực và ảo cho WRB cartrridges thật tốt như là số của WRBXs

II.4.3 PL/SQL Agent Pages

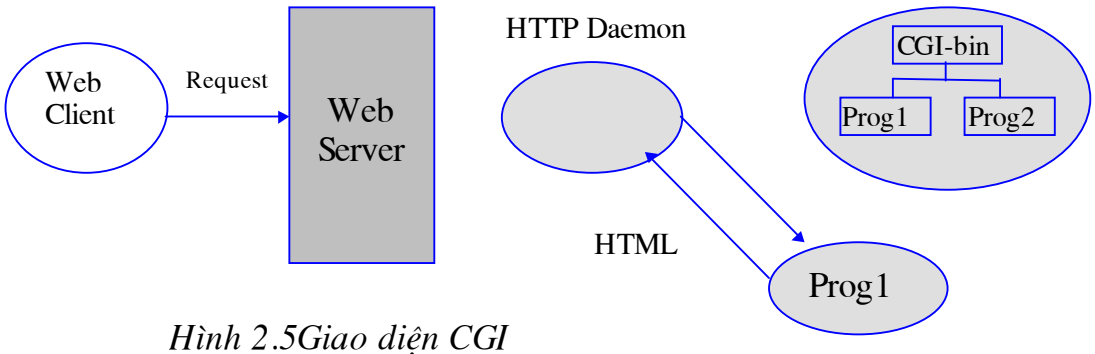
Việc quản lý PL/SQL Agent có nghĩa là quản lý việc mô tả kết nối cơ sở dữ liệu (Database Connection Descriptors(DCDs) sử dụng để kiến thiết nó và chỉ rõ khi kết hợp với cơ sở dữ liệu.

II.4.4 Oracle7 Server Manager

Oracle7 Server là một sản phẩm tinh vi hơn. Nó thích hợp hơn cho việc quản lý sử dụng Oracle Server hay một cách trực tiếp thông qua ngôn ngữ SQL.

II.5 Giao diện CGI

Khi Web Listener nhận một địa chỉ URL như một câu hỏi thực hiện một ứng dụng CGI. Web Listener thông qua URL tới những tiến trình này và kết hợp với nó thông qua chuẩn vào và chuẩn ra, bởi vậy tiến trình CGI có thể lấy dữ liệu nhập vào nếu cần từ URL hay từ chuẩn vào (standard Input). Và sẽ gửi chúng ra Web Listener thông qua chuẩn ra (standard output) và Web Listener gửi cho Browser. (Hình 2.5)



Hình 2.5Giao diện CGI

II.6 PL/SQL Agent

Một WRBX sử dụng PL/SQL kết nối ngay lập tức với cơ sở dữ liệu khi nó đã được tạo và đợi một câu hỏi đưa vào.Điều nàycho phép yêu cầu được tiến hành một cách nhanh chóng hơn. Lược đồ cơ sở dữ liệu cho phép WRBX kết nối và xác định rõ trong cấu hình File.

II.7 Xác định và Sử dụng PL/SQL

Nếu một URL đưa một câu hỏi chứa đựng sâu "OWA". Listener hoàn thiện câu hỏi sử dụng PL/SQL Agent. Công việc này sẽ thông qua WRB hay CGI phụ thuộc vào người quản trị WebServer đã đặt cấu hình "chỉ dẫn ánh xạ" (directory mappings) như thế nào. Khi đó PL/SQL Agent sẽ thực hiện ứng dụng viết bởi PL/SQL và trả lại dạng HTML cho WebListener đưa ra trang Web.

Khi một URL chứa đựng PL/SQL Agent, nó bao gồm một DCD (Database Connection Descriptor). DCD xác định cả hai đặc quyền truy nhập cơ sở dữ liệu PL/SQL Agent khi thực hiện yêu cầu và lược đồ hoá.

Các thủ tục PL/SQL tồn tại trong cơ sở dữ liệu. PL/SQL Agent bao hàm câu lệnh được đưa tới cơ sở dữ liệu, Sau đó tiến hành thực hiện và gửi ra ngoài và thông báo trạng thái trở lại cho PL/SQL Agent.

B. XÂY DỰNG CHƯƠNG TRÌNH TRUY NHẬP CƠ SỞ DỮ LIỆU THEO GIAO DIỆN CGI

Như đã nói ở trên chương trình CGI (Common GateWay Interface) là công nghệ chuẩn được sử dụng bởi một Web Listener dùng HTTP Server để thực hiện một chương trình sinh ra tài liệu dạng HTML. Ví dụ ta có thể viết một chương trình CGI để thực hiện việc lưu trữ và lấy dữ liệu từ một hệ cơ sở dữ liệu bất kỳ dưới nhiều dạng khác nhau kể cả dưới dạng nhị phân (file ảnh) tức là hoàn toàn có thể thao tác với cơ sở dữ liệu thông qua Web.

Cụ thể với hệ cơ sở dữ liệu ORACLE cho phép mỗi User có thể kết nối với CSDL bằng chính tên mình đã đăng ký hoặc chạy PL/SQL và giao diện với Oracle7 Server. Đặc biệt là chúng ta có thể viết một chương trình CGI bằng nhiều ngôn như C/C++, COBOL. . . mà qua Web ta có thể Select, Insert, Update . . dữ liệu từ một Table nào đó trong cơ sở dữ liệu. Những chương trình CGI như vậy được gọi là OCI (Oracle Call Interface). Để viết một chương trình OCI có thể tiến hành theo các bước như sau:

- Xác định cấu trúc dữ liệu cho phép kết nối vào Oracle Server nào hay cơ sở dữ liệu nào.
- Kết nối vào một hay nhiều cơ sở dữ liệu Oracle.
- Mở một hay nhiều tiến trình SQL cần thiết cho chương trình.
- Xác định nhiệm vụ của SQL hay PL/SQL cho chương trình.
- Đóng các Cursors

- Huỷ bỏ kết nối từ cơ sở dữ liệu.

Tuy nhiên nó có nhược điểm nhỏ là ngữ trình thông qua chuẩn CGI do dùng các biến môi trường nên thực thi chậm. Nhưng lại có ưu điểm là khi chạy đưa ra kết quả là tài liệu HTML chuẩn. Để khắc phục nhược điểm đó người ta đã đưa ra giải pháp là dùng OWA (Oracle Web Agent)

I OWA - ORACLE WEB AGENT

I.1 Oracle Web Agent là gì

OWA là những chương trình con được xây dựng thành thủ tục, hàm mang chức năng khác nhau trong PL/SQL của Oracle. Dùng OWA để biến câu hỏi của User thông qua các Store Procedure chuyển thành trang Web và trả lại kết quả. Để hiểu được tính năng cũng như nhiệm vụ của OWA trước hết chúng ta xem xét hai khái niệm HTP (Hypertext Procedure) và HTF (Hypertext Function).

I.2 Hypertext Procedure (HTP)

Một HTP được sinh ra là "một dòng" trong tài liệu HTML có chứa đựng những thẻ HTML. Ví dụ Htp.anchor là thủ tục sinh ra một anchor tag. HTP phần lớn sẽ sử dụng những OWA.

I.3 Hypertext Function

Một HTF trả lại những thẻ HTML tương ứng với chính tên của nó. Tuy nhiên nó không thích đáng được gọi là một HTF bởi vì thẻ HTML không thông qua PL/SQL Agent. Đầu ra của một HTF phải thông qua HTP.printf được sắp xếp một phần trong tài liệu HTML

Mọi HTF đều tương ứng với một HTP. Mặc dù vậy HTF được sinh ra chỉ khi người lập trình cần gọi đến, ví dụ :

htp.header(1,htf.italic('Title'));

Với dòng lệnh trên htf.italic sẽ cho ta sâu ký tự <*I*>*Title*</*I*> và sau khi thông qua htp.header thì sâu ký tự được sinh ra trong tài liệu HTML sẽ có dạng như sau:

<H1><I>Title</I></H1>

I.4 Các OWA cơ bản

I.4.1 OWA_UTIL (owa_utilities)

Là tập hợp của đầy đủ tiện ích thủ tục để xây dựng HTF & HTP. Tuỳ theo mục đích mà người lập trình có thể sử dụng hàm hoặc thủ tục nào chẳng hạn có thể dùng hàm *OWA_util.get_cgi_env(param_name in Varchar2)* để xác định biến môi trường CGI đã dùng trong chương trình, hoặc có thể dùng thủ tục *OWA_util.showpage* để xác định đầu ra HTML của một thủ tục PL/SQL gọi từ SQL*PLUS hay SQL*DBA,. . .

I.4.2 OWA_PATTERN (Pattern Matching Utilities)

OWA_pattern cung cấp cho chúng ta 3 hoạt động sau đây:

+ MATCH: Xác định rõ một biểu thức đã tồn tại trong một xâu. Đây là một hàm trả lại giá trị TRUE hay FALSE

+ AMATCH: Đây là hàm trả lại giá trị nguyên và kết thúc một xâu mà biểu thức thường đã tìm thấy. Nếu biểu thức không tìm thấy sẽ trả lại giá trị là 0

+ CHANGE: Cho phép thay thế (cập nhật) phần chia của xâu đã được Matched với một biểu thức thông thường và xâu mới. CHANGE có thể là một thủ tục hay một hàm. Nếu là hàm thì trả lại thời gian tìm thấy và thay thế

- OWA_TEXT (Text Manipulation Utilities)

OWA _text được sử dụng chủ yếu bởi OWA _pattern nhưng hàm là "ngoại hiện" mà chúng ta có thể sử dụng chúng một cách trực tiếp nếu đã hoàn toàn đồng ý. Ví dụ có thể có thể dùng OWA_text để chuyển đổi một xâu dài thành nhiều dòng hoặc có thể thêm nội dung vào một dòng,. . .

I.4.3 OWA_COOKIE (Cookie Utilities)

Là một gói bao bọc đầy đủ ta có thể gửi và lấy cookies từ Client, Cookie không rõ ràng đối với Client. Nó duy trì trạng thái thông qua phiên làm việc của Client. Ta có thể chuyển đổi thông tin từ dạng xâu sang một Cookie nếu sử dụng hàm OWA_cookie.get(name), . . .

I.4.4 OWA_INIT

Đây là gói chứa đầy đủ mọi thông tin về thời gian. Chẳng hạn ta có thể đặt trước thời gian sử dụng Cookie với giờ quy định GMT (Greenwich Mean Time). Cookie sẽ chỉ sử dụng đúng khoảng thời gian đã được định nghĩa. Nếu không ở trong múi giờ GMT thì có thể đưa vùng thời gian sử dụng vào.

I.5 Xây dựng chương trình

Thông thường với một hệ cơ sở dữ liệu nếu chúng ta muốn thao tác được với dữ liệu trong hệ cơ sở dữ liệu thì chúng ta phải trực tiếp sử dụng hệ cơ sở dữ liệu đó. Chẳng hạn nếu muốn thay đổi dữ liệu từ một Table trong Hệ quản trị cơ sở dữ liệu Oracle thì chúng ta phải trực tiếp tác động vào Table đó thông qua ngôn ngữ SQL (Structure Query Language). Nhưng thay vì công việc là phải nhập dữ liệu trực tiếp vào Table bằng câu lệnh *Insert,* hay xem dữ liệu bằng câu lệnh *Select* trong môi trường ngôn ngữ SQL, thì ta có thể xâm nhập vào cơ sở dữ liệu để thao tác với cơ sở dữ liệu đó trên Web. Thông qua Web người sử dụng không cần biết mình đang sử dụng hệ cơ sở dữ liệu nào,

và nó thực hiện như thế nào nhưng vẫn đảm bảo đáp ứng đúng nhu cầu. Chẳng hạn với chương trình FULL_TEXT (được xây dựng tại CSE): Là chương trình Tra cứu nội dung các văn bản cho Bộ Ngoại Giao, được xây dựng năm 1997 trên môi trường ORACLE. Chương trình cho phép truy nhập đến nội dung các văn bản lưu giữ trong Database của Oracle, tìm kiếm trong nội dung của toàn bộ các văn bản các từ, cụm từ và sau đó cho phép người dùng có thể hiển thị đầy đủ toàn bộ nội dung các văn bản tìm được trên Web.

Đối với hệ quản trị cơ sở dữ liệu ORACLE, khi người sử dụng nhập dữ liệu thông qua Form giao diện, Web Browser trình diện yêu cầu đó lên Oracle Web Server. Web Listener có nhiệm vụ "nghe" và tiếp nhận yêu cầu URL gửi vào từ đâu thông qua cổng giao diện nào, sau đó sẽ xác định dịch vụ yêu cầu và gửi tới WRB (Web Request Broker). WRB gửi yêu cầu đó tới các Cartridger như PL/SQL, JAVA và WRBXs (Web Request Broker) gọi thực hiện tiến trình CGI. Sau khi thực hiện xong tiến trình CGI trả lại kết quả dữ liệu dưới dạng mã HTML chuẩn. WRB gửi kết quả đó tới Web Listener, Web Listener gửi trả Web Browser, quá trình kết thúc.

Sau đây là chương trình minh hoạ, chương trình được xây dựng nhằm thể hiện việc thông qua Web người sử dụng tác động như thế nào tới cơ sở dữ liệu. Chương trình có sử dụng những OWA cơ bản, và Table *ngay_sinh* trong Database DU/DU@STU. Chương trình bao gồm 1 Package **demo1** với 7 thủ tục sau:

- Thủ tục thứ nhất **nhap_dk** đảm nhiệm chức năng tạo một Form giao diện để người dùng nhập dữ liệu yêu cầu và trình diện yêu cầu lên Oracle Web Server. Sau khi trình diện lên Server thủ tục **hien_kq** sẽ được gọi bằng câu lệnh:

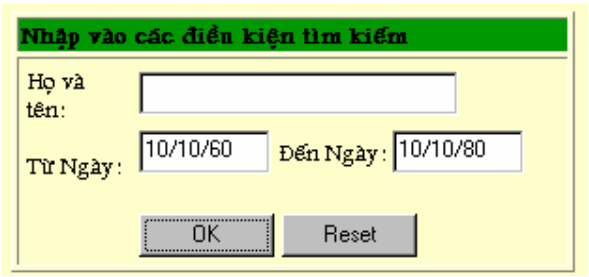
htp.print('<Form action="http://acernt:800/du/owa/demo1.hien_kq">');

Khi nhận được yêu cầu Web Listener sẽ "nghe" yêu cầu và gửi tới Web Request Broker. Web Request Broker gọi đến Cartridger SQL và tìm kiếm Table. Khi đã tìm thấy sẽ tiến hành thực hiện nhiệm vụ tìm kiếm theo yêu cầu và trả lại kết quả:

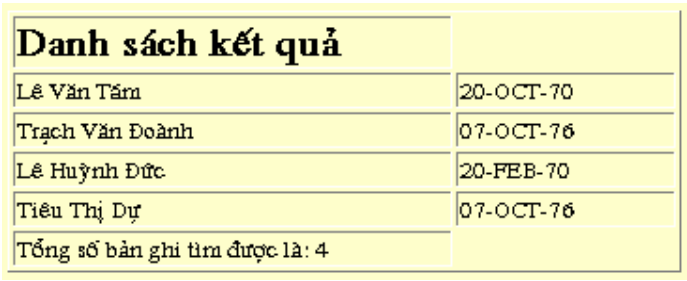
```
if para is not null then
     para := 'select hoten, NS from Ngay_sinh Tab1 where ' ||
para;
end if;
if para is null then
     para:='select hoten, NS from ngay_sinh Tab1';
end if;
c1:=dbms_sql.open_cursor;
     dbms_sql.parse(c1,para, dbms_sql.v7);
     dbms_sql.define_column(c1,1,ho_ten, 30);
     dbms_sql.define_column(c1,2,ngay_sinh);
     status := dbms_sql.execute(c1);
loop
     if dbms_sql.fetch_rows(c1) >0 then
               ts:=ts+1;
               dbms_sql.column_value(c1,1,ho_ten);
               dbms_sql.column_value(c1,2,ngay_sinh);
                    htp.print('<tr>');
          htp.print('<td>'||Ho_ten|| </td><td>'||ngay_sinh||
</td>');
                    htp.print('</tr>');
     else exit;
     end if;
end loop;
```

Sau khi thực hiện xong thủ tục **hien_kq** đưa ra kết quả dưới dạng mã HTML chuẩn. Dịch vụ WRB Service sẽ nhận kết quả và gửi trả Web Listener. Web Listener báo tín hiệu hoàn thành và gửi trả Web Browser.

Ví dụ ta muốn xem tất cả những người sinh từ ngày 10/10/60 đến ngày 10/10/80. Nhập điều kiện.



Sau khi nhập vào điều kiện xem xét và chọn nút OK ta được kết quả trả về trên Web Browser như sau:



Danh sách kết quả	
Lê Văn Tám	20-OCT-70
Trạch Văn Đoành	07-OCT-76
Lê Huỳnh Đức	20-FEB-70
Tiêu Thị Dự	07-OCT-76
Tổng số bản ghi tìm được là: 4	

- Thủ tục **test** có chức năng cho người dùng xem toàn bộ dữ liệu có trong cơ sở dữ liệu. Khi Web Browser trình diện yêu cầu tới Web Server. Web Listener "nghe" yêu cầu và gửi đến WRB, sau khi WRBXs thực hiện xong tiến trình CGI gửi trả kết quả là câu lệnh tới WRB Cartridger PL/SQL

*Select * from ngay_sinh;*

Sau khi thực hiện xong kết quả hiện lên Web Browser như sau:

Lê Văn Tám	20-OCT-70
Trần Trung Hiếu	01-JAN-50
Trạch Văn Đoành	07-OCT-76
Lê Huỳnh Đức	20-FEB-70
Tiêu Thị Dự	07-OCT-76

- Thủ tục thứ ba **Form_nhap** có chức năng tạo một Form giao diện nhận thông tin của người dùng. Khi thủ tục này được gọi nó sẽ gọi tiếp đến thủ tục **insert_data**

htp.print('<Form action="http://acernt:800/du/owa/demo1.insert_data">');

Thủ tục **insert_data** đảm nhận chức năng tiếp nhận thông tin người sử dụng trình diện lên từ **Form_nhap** và nhập dữ liệu vào cơ sở dữ liệu.

```
begin
     htp.print('<html>');
     htp.print('<body>');
     insert      into      DU.ngay_sinh           values(ten,
to_date(ngay,'dd/mm/yy'));
     htp.print('<b>§· Insert D÷ LiÖu Vµo Table </b>');
     htp.print('</body>');
     htp.print('</html>');
end;
```

Sau khi thủ tục này thực hiện Cartridger SQL nhận dược câu lệnh:

insert into DU.ngay_sinh values(ten, to_date(ngay,'dd/mm/yy'));

Ví dụ muốn nhập thêm dữ liệu vào Table ta chỉ việc nhập vào Form giao diện:

Sau khi đã nhập dữ liệu nhấn nút OK thì dữ liệu được nhập vào Table chỉ định, kết quả trả về trên Web Browser như sau:

Lê Văn Tám	20-OCT-70
Trần Trung Hiếu	01-JAN-50
Trạch Văn Đoành	07-OCT-76
Nguyễn Hữu Thắng	30-MAY-80
Lê Huỳnh Đức	20-FEB-70
Tiêu Thị Dự	07-OCT-76

- Thủ tục **nhap_dkx** đảm nhận chức năng tạo một form giao diện để người dùng nhập thông tin cần thiết để xoá dữ liệu theo điều kiện. Khi Web Browser trình diện yêu cầu lên Web Server thủ tục **hien_kqx** sẽ được gọi.

htp.print('<Form action="http://acernt:800/du/owa/demo1.hien_kqx">');

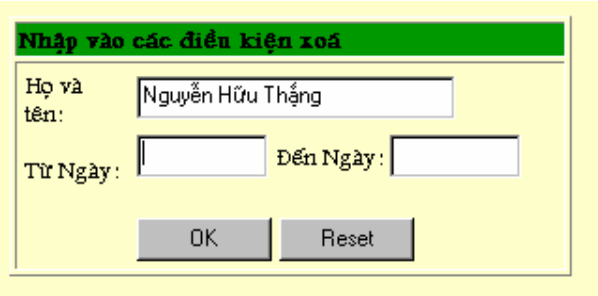
Thủ tục này có chức năng tiếp nhận thông tin nhập vào từ form được tạo ra trong thủ tục **nhap_dkx** và xoá dữ liệu theo đúng yêu cầu nhận được. Khi tiến trình CGI **nhap_dkx** hoạt động sẽ gọi đến Cartridger SQL:

```
if para is not null then
     para := 'delete Ngay_sinh tab  where ' || para;
end if;
if para is null then
     para:='delete ngay_sinh ';
end if;
   cursor_name := DBMS_SQL.OPEN_CURSOR;
   DBMS_SQL.PARSE(cursor_name, para, DBMS_SQL.V7);
   ret := DBMS_SQL.EXECUTE(cursor_name);
   DBMS_SQL.CLOSE_CURSOR(cursor_name);
```

Sau khi thực hiện xong tiến trình CGI này Cartridger PL/SQL nhận được câu lệnh:

```
     delete Ngay_sinh tab  where ' || para;
```

Ví dụ có thể xoá bất kỳ một row nào đó theo điều kiện nhập. Chẳng hạn ta muốn xoá một người có tên Nguyễn Hữu Thắng



Sau khi nhấn nút OK Cartridger PL/SQL sẽ nhận được câu lệnh

Delete ngay_sinh tab where hoten='Nguyễn Hữu Thắng';

dữ liệu trong Table sẽ bị xoá theo điều kiện họ tên là La Văn Cầu. Kết quả như sau:

Lê Văn Tám	20-OCT-70
Trần Trung Hiếu	01-JAN-50
Trạch Văn Đoành	07-OCT-76
Lê Huỳnh Đức	20-FEB-70
Tiêu Thị Dự	07-OCT-76

KẾT LUẬN

Trong thời gian làm luận văn em đã tìm hiểu và nghiên cứu được một số vấn đề hệ quản trị cơ sở dữ liệu Oracle với Oracle Web Server, hệ thống Web nói chung và dịch vụ Web trên mạng. Từ đó tìm hiểu cách thức khai thác cơ sở dữ liệu thông qua Web. Cách thức CGI truy nhập CSDL, đặc điểm cơ bản của một chương trình CGI cũng như phân tích cách thức hoạt động của một chương trình CGI và ứng dụng của nó trong hệ cơ sở dữ liệu Oracle. Xây dựng chương trình truy nhập cơ sở dữ liệu bằng nhôn ngữ C, và chương trình CGI truy nhập CSDL ORACLE .

Vì điều kiện thời gian có hạn nên luận văn chỉ dừng ở mức nghiên cứu cách thức truy nhập cơ sở dữ liệu bằng chương trình ngoài CGI và đưa ra những ví dụ minh hoạ đơn giản. Trong thời gian tiếp theo em sẽ tiếp tục nghiên cứu thêm một số phương pháp khác trợ giúp Web Server khai thác cơ sở dữ liệu như phương pháp ISAPI, ASP hay JAVA nhằm đáp ứng tối đa yêu cầu của người sử dụng và xây dựng những ứng dụng cụ thể.

Một lần nữa em xin chân thành cám ơn toàn thể các thày cô giáo khoa CNTT và toàn thể nhân viên công ty CSE.

Hà nội - 1998

Người thực hiện

Tiêu Thị Dự

PHỤ LỤC

Phụ lục 1 Chương trình nguồn ktra.c xử lý Form được viết bằng ngôn ngữ C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
	char	InputBuffer[4096] ;
	typedef struct field_s
{
	char			*f_name ;
	char			*f_value ;
	struct field_s		*f_next ;
}
	field_t, *pfield_t ;
	field_t			*field_list = NULL ;
void strcvrt( char * cStr, char cOld, char cNew )
{
	int i = 0 ;
	while ( cStr[i] )
	{
		if ( cStr[i] == cOld )
			cStr[i] = cNew ;
		i++ ;
	}
}
int TwoHex2Int( char *pC )
{
	int	Hi, Lo, Result=0 ;
	Hi = pC[0] ;
	if ( '0' <= Hi && Hi <= '9' )
		Hi -= '0' ;
	else if ( 'a' <= Hi && Hi <= 'f' )
		Hi -= ('a' - 10) ;
	else if ( 'A' <= Hi && Hi <= 'F' )
		Hi -= ('A' - 10) ;
	Lo = pC[1] ;
	if ( '0' <= Lo && Lo <= '9' )
		Lo -= '0' ;
```

```
        else if ( 'a' <= Lo && Lo <= 'f' )
                Lo -= ('a' - 10) ;
        else if ( 'A' <= Lo && Lo <= 'F' )
                Lo -= ('A' - 10) ;
        Result = Lo + 16*Hi ;
        return(Result) ;
}
void urlDecode( char *p )
{
        char   *pD = p ;
        while (*p)
        {
                if ( *p == '%' )
                {
                        p++ ;
                        if( isxdigit(p[0]) && isxdigit(p[1]))
                        {
                                *pD++ = (char) TwoHex2Int(p) ;
                                 p += 2 ;
                        }
                }
                else
                {
                        *pD++ = *p++ ;
                }
        }
        *pD = '\0' ;
}
        field_t *f_newitem(char*f_name, char *f_value)
{
        field_t         *f_tmp = NULL ;
        if((f_tmp=(field_t*)malloc(sizeof(field_t)))== NULL )
                return(NULL);
        if((f_tmp->f_name=(char*)malloc(strlen(f_name) ))== NULL )
                return(NULL) ;
        if((f_tmp->f_value=(char*)malloc( strlen(f_value) ) ) == NULL )
                return(NULL) ;
        strcpy( f_tmp->f_name, f_name ) ;
        strcpy( f_tmp->f_value, f_value ) ;
        f_tmp->f_next = NULL ;
        return(f_tmp) ;
```

```
}
        char *f_value( char *f_name )
{
        field_t         *f_tmp = NULL ;

        if(f_name == NULL )
                return(NULL) ;
        f_tmp = field_list ;
        while (f_tmp)
        {
                if ( stricmp(f_tmp->f_name, f_name ) == 0 )
                        return(f_tmp->f_value) ;
                f_tmp = f_tmp->f_next ;
        }
        return(NULL) ;
}
void f_additem( field_t *f_item )
{
        if ( f_item == NULL )
                return ;
        if ( field_list == NULL )
        {
                field_list = f_item ;
                return ;
        }
        f_item->f_next = field_list ;
        field_list = f_item ;
}
void StoredField( char *item )
{
        char *p = NULL ;
        char *field_name = NULL, *field_value = NULL ;
        if ( item == NULL || *item == '\0' )
                return ;
        p = strchr( item, '=' ) ;
        *p++ = '\0' ;
        urlDecode(item) ;
        urlDecode(p) ;
        strcvrt( p, '\n', ' ' ) ;
        strcvrt( p, '+', ' ') ;
        f_additem( f_newitem(item, p) ) ;
```

```
}
int main()
{
        field_t         *f_tmp = NULL ;
        int     ContentLength ;
        int             x, i ;
        char    *p, *pRequestMethod = NULL ;
        printf("Content-Type: text/html\n\n") ;
        printf("<HTML>\n<HEAD>\n<title>CGI
Dump</title>\n</HEAD>\n") ;
        printf("<BODY><h1>CGI Dump</h1>\n<hr>\n") ;
        printf("<table>width=\"100%%\"  cellspacing=\"0\"  cellpadding=\"0\"
border=\"0\">\n") ;
        printf("<tr>\n    <td width=\"30%%\" align=\"left\" valign=\"top\">\n")
;
        printf("Content_Length") ;
        printf("\n          </td>\n          <td   width=\"70%%\"   align=\"left\"
valign=\"top\">\n") ;
        p = getenv("CONTENT_LENGTH") ;
        if ( p != NULL && *p != '\0' )
                printf(p) ;
        else
                printf(" ") ;
        printf("\n </td>\n</tr>\n") ;
        printf("<tr>\n<td width=\"30%%\" align=\"left\" valign=\"top\">\n") ;
        printf("Content_Type") ;
        printf("\n     </td>\n          <td>     width=\"70%%\"     align=\"left\"
valign=\"top\">\n") ;
        p = getenv("CONTENT_TYPE") ;
        if ( p != NULL && *p != '\0' )
                printf(p) ;
        else
                printf(" ") ;
        printf("\n    </td>\n</tr>\n") ;
        printf("<tr>\n    <td width=\"30%%\" align=\"left\" valign=\"top\">\n")
;
        printf("Gateway_Interface") ;
        printf("\n          </td>\n          <td   width=\"70%%\"   align=\"left\"
valign=\"top\">\n") ;
        p = getenv("GATEWAY_INTERFACE") ;
        if ( p != NULL )
```

```
                printf(p) ;
        else
                printf(" ") ;
        printf("\n    </td>\n</tr>\n") ;
        printf("<tr>\n    <td width=\"30%%\" align=\"left\" valign=\"top\">\n")
;
        printf("http_accept") ;
        printf("\n            </td>\n            <td   width=\"70%%\"   align=\"left\"
valign=\"top\">\n") ;
        p = getenv("HTTP_ACCEPT") ;
        if ( p != NULL && *p != '\0' )
                printf(p) ;
        else
                printf(" ") ;
        printf("\n    </td>\n</tr>\n") ;
        printf("<tr>\n    <td width=\"30%%\" align=\"left\" valign=\"top\">\n")
;
        printf("http_referer") ;
        printf("\n            </td>\n            <td   width=\"70%%\"   align=\"left\"
valign=\"top\">\n") ;
        p = getenv("HTTP_REFERER") ;
        if ( p != NULL && *p != '\0' )
                printf(p) ;
        else
                printf(" ") ;
        printf("\n    </td>\n</tr>\n") ;
        printf("<tr>\n    <td width=\"30%%\" align=\"left\" valign=\"top\">\n")
;
        printf("path_info") ;
        printf("\n            </td>\n            <td   width=\"70%%\"   align=\"left\"
valign=\"top\">\n") ;
        p = getenv("PATH_INFO") ;
        if ( p != NULL && *p != '\0' )
                printf(p) ;
        else
                printf(" ") ;
        printf("\n    </td>\n</tr>\n") ;

        printf("<tr>\n    <td width=\"30%%\" align=\"left\" valign=\"top\">\n")
;
        printf("query_string") ;
```

```
        printf("\n          </td>\n          <td  width=\"70%%\"  align=\"left\"
valign=\"top\">\n") ;
        p = getenv("QUERY_STRING") ;
        if ( p != NULL && *p != '\0' )
                printf(p) ;
        else
                printf(" ") ;
        printf("\n    </td>\n</tr>\n") ;
        printf("<tr>\n    <td width=\"30%%\" align=\"left\" valign=\"top\">\n")
;
        printf("remote_addr") ;
        printf("\n          </td>\n          <td  width=\"70%%\"  align=\"left\"
valign=\"top\">\n") ;
        p = getenv("REMOTE_ADDR") ;
        if ( p != NULL && *p != '\0' )
                printf(p) ;
        else
                printf(" ") ;
        printf("\n    </td>\n</tr>\n") ;
        printf("<tr>\n    <td width=\"30%%\" align=\"left\" valign=\"top\">\n")
;
        printf("request_method") ;
        printf("\n          </td>\n          <td  width=\"70%%\"  align=\"left\"
valign=\"top\">\n") ;
        p = getenv("REQUEST_METHOD") ;
        if ( p != NULL && *p != '\0' )
                printf(p) ;
        else
                printf(" ") ;
        printf("\n    </td>\n</tr>\n") ;
        printf("<tr>\n    <td width=\"30%%\" align=\"left\" valign=\"top\">\n")
;
        printf("script_name") ;
        printf("\n          </td>\n          <td  width=\"70%%\"  align=\"left\"
valign=\"top\">\n") ;
        p = getenv("SCRIPT_NAME") ;
        if ( p != NULL && *p != '\0' )
                printf(p) ;
        else
                printf(" ") ;
        printf("\n    </td>\n</tr>\n") ;
```

```
        printf("<tr>\n    <td width=\"30%%\" align=\"left\" valign=\"top\">\n")
;
        printf("server_name") ;
        printf("\n          </td>\n          <td   width=\"70%%\"   align=\"left\"
valign=\"top\">\n") ;
        p = getenv("SERVER_NAME") ;
        if ( p != NULL && *p != '\0' )
                printf(p) ;
        else
                printf(" ") ;
        printf("\n    </td>\n</tr>\n") ;
        printf("<tr>\n    <td width=\"30%%\" align=\"left\" valign=\"top\">\n")
;
        printf("server_port") ;
        printf("\n          </td>\n          <td   width=\"70%%\"   align=\"left\"
valign=\"top\">\n") ;
        p = getenv("SERVER_PORT") ;
        if ( p != NULL && *p != '\0' )
                printf(p) ;
        else
                printf(" ") ;
        printf("\n    </td>\n</tr>\n") ;
        printf("<tr>\n    <td width=\"30%%\" align=\"left\" valign=\"top\">\n")
;
        printf("server_protocol") ;
        printf("\n          </td>\n          <td   width=\"70%%\"   align=\"left\"
valign=\"top\">\n") ;
        p = getenv("SERVER_PROTOCOL") ;
        if ( p != NULL && *p != '\0' )
                printf(p) ;
        else
                printf(" ") ;
        printf("\n    </td>\n</tr>\n") ;
        printf("<tr>\n    <td width=\"30%%\" align=\"left\" valign=\"top\">\n")
;
        printf("server_software") ;
        printf("\n          </td>\n          <td   width=\"70%%\"   align=\"left\"
valign=\"top\">\n") ;
        p = getenv("SERVER_SOFTWARE") ;
        if ( p != NULL && *p != '\0' )
                printf(p) ;
```

```
        else
                printf(" ") ;
        printf("\n    </td>\n</tr>\n") ;
        printf("</table>\n") ;
        setvbuf( stdin, NULL,_IONBF, 0 ) ;
        pRequestMethod = getenv("REQUEST_METHOD") ;
        if ( pRequestMethod == NULL || pRequestMethod[0] == '\0' )
        {
                printf("\nERROR:Request Method error\n") ;
                goto error ;
        }
        if ( strcmp( pRequestMethod, "POST" ) == 0 )
        {
                p = getenv("CONTENT_LENGTH") ;
                if ( p != NULL )
                        ContentLength = atoi(p) ;
                else
                        ContentLength = 0 ;
                if ( ContentLength > sizeof(InputBuffer) - 1 )
                        ContentLength = sizeof(InputBuffer) - 1 ;

                i = 0 ;
                while ( i < ContentLength )
                {
                        x = fgetc(stdin) ;
                        if ( x == EOF )
                                break ;
                        InputBuffer[i++] = x ;
                }
                InputBuffer[i] = '\0' ;
                ContentLength = i ;
                if ( InputBuffer != NULL && *InputBuffer != '\0' )
                        printf("\n<p><p>\n<b>InputBuffer:        </b>        %s\n",
InputBuffer ) ;
                p = getenv("CONTENT_TYPE") ;
                if ( p == NULL )
                {
                        printf("\nERROR: content_type error.\n") ;
                        goto error ;
                }
                if ( strcmp( p, "application/x-www-form-urlencoded" ) == 0 )
```

```
                {
                        p = strtok( InputBuffer, "&" ) ;
                        while ( p != NULL )
                        {
                                StoredField( p ) ;
                                p = strtok( NULL, "&" ) ;
                        }
                }
        }
        if ( field_list == NULL )
        {
                printf("\n<h2>No variables</h2>\n") ;
                goto error ;
        }
        printf("<hr>\n<h2>Variables</h2>\n") ;
        printf("<table  width=\"100%%\"  cellspacing=\"0\"  cellpadding=\"0\"
border=\"0\">\n") ;
        f_tmp = field_list ;
        while (f_tmp)
        {
                printf("<tr>\n                <td    width=\"30%%\"    align=\"left\"
valign=\"top\">\n") ;
                printf(f_tmp->f_name) ;
                printf("          </td>\n          <td  width=\"70%%\"  align=\"left\"
valign=\"top\">\n") ;
                printf(f_tmp->f_value) ;
                printf("    </td>\n</tr>\n") ;
         f_mp = f_tmp->f_next ;
        }
        printf("</table>\n</BODY>\n</HTML>") ;
        return(0) ;
error:
        printf("\n</BODY>\n</HTML>") ;
        return(-1) ;
}
```

Phụ lục 2 Chương trình nguồn viết bằng PL/SQL dựa trên OWA (Oracle Web Agent).

```
drop package demo1;

create package demo1  as

procedure start_pro;
procedure  nhap_dk;
procedure  hien_kq(ten varchar2  default null,  tu_ngay varchar2  default null,
den_ngay varchar2  default null);

procedure form_nhap (ten varchar2 default null, ngay  varchar2 default null);
procedure insert_data(ten varchar2  default null, ngay varchar2  default null) ;

procedure nhap_dkx;
procedure  hien_kqx(ten varchar2  default null,  tu_ngay varchar2  default
null, den_ngay varchar2  default null) ;
end demo1;
/
-----------------------------------------------------------------
create package body demo1 as
procedure nhap_dk is
begin
htp.print('<html>');
htp.print('<Body>');
htp.print('<Form action="http://acernt:800/du/owa/demo1.hien_kq">');
htp.print('<center>');
htp.nl;
htp.nl;
htp.print('<table width="50%" border=1>');
htp.print('<tr><td bgcolor="0069060">');
htp.bold(' Nhập vào các điều kiện tìm kiếm');
htp.print('</td></tr>');
htp.print('<tr><td>');
htp.print('<table width ="95%" border=0>');
htp.print('<tr><td>');
htp.print('Họ và tên:');
htp.print('</td><td>');
htp.print('<input name ="ten" type= text  size=30 >');
htp.print('</td></tr>');

htp.print('<tr><td>');
htp.print('Từ Ngày:');
```

```
htp.print('</td><td>');
htp.print('<input name ="tu_ngay" type= text  size=10 >');
htp.print('Đến Ngày:');
htp.print('<input name ="den_ngay" type= text  size=10 >');
htp.print('</td></tr>');
htp.print('<tr></tr>');

htp.print('<tr><td><br></td><td>');
htp.print('<input type="Submit"  value="OK"> ');
htp.print('<input type="Reset"  value="Reset"> ');
htp.print('</td></tr>');
htp.print('</table>');

htp.print('</td></tr>');
htp.print('</table>');


htp.print('</center>');
htp.print('</form >');

htp.print('</body>');
htp.print('</html>');

end;
----------------------------------
procedure  hien_kq(ten varchar2  default null,  tu_ngay varchar2  default null,
den_ngay varchar2  default null) is

c1      integer;
status  integer;
ngay1  date;
ngay2  date;
ngay_sinh  date;
ho_ten   varchar2(30);
para    varchar2(1000):=null;
dem    integer := null;
ts      integer:=0;
begin
htp.print('<html>');
htp.print('<Body bgcolor="FFFFDC">');
htp.print('<center>');
```

```
htp.print('<table width= "60%"  border=1>');
htp.print('<tr><td>');
htp.print('<h2>Danh sách kết quả </h2>');


if tu_ngay  is not null then para:=para|| ' tab1.NS
>='''||to_date(tu_ngay,'DD/MM/YY')||''''; end if;
if den_ngay is not null then para:=para||'$'|| ' tab1.NS
<='''||to_date(den_ngay,'DD/MM/YY')||''''; end if;

if ten is not null then para:= para||'$'||'tab1.hoten='''|| ten||''''   ; end if;

dem:=length(para);
if (SUBSTR(para,1,1)='$') then para:=SUBSTR(para,2,dem-1); end if;
if (SUBSTR(para,dem,1)='$') then para:=SUBSTR(para,1,dem-1); end if;
para:= replace(para,'$',' AND ');

if para is not null then
        para := 'select hoten, NS from Ngay_sinh Tab1 where ' || para;
end if;

if para is null then
        para:='select hoten, NS from ngay_sinh Tab1';
end if;


c1:=dbms_sql.open_cursor;
        dbms_sql.parse(c1,para, dbms_sql.v7);
        dbms_sql.define_column(c1,1,ho_ten, 30);
        dbms_sql.define_column(c1,2,ngay_sinh);
        status := dbms_sql.execute(c1);

loop
        if dbms_sql.fetch_rows(c1) >0 then
                        ts:=ts+1;
                        dbms_sql.column_value(c1,1,ho_ten);
                        dbms_sql.column_value(c1,2,ngay_sinh);

                                htp.print('<tr>');
                                htp.print('<td>'|| Ho_ten||
'</td><td>'||ngay_sinh||'</td></tr>');
```

```
        else exit;
        end if;
end loop;
if ts <1 then
        htp.print('Không tìm thấy bản ghi nào thoả mãn điều  kiện ');
else
        htp.print('Tổng số bản ghi tìm được là: '||ts);
end if;


htp.print('</table>');
htp.print('</body>');
htp.print('</html>');
end;

-------------------------------------
procedure form_nhap (ten varchar2 default null, ngay  varchar2 default null)
is
begin
htp.print('<html>');
htp.print('<Body>');
htp.print('<Form action="http://acernt:800/du/owa/demo1. insert_data">');

htp.print('<center>');

htp.nl;
htp.nl;

htp.print('<table width="50%" border=1>');
htp.print('<tr><td bgcolor="0069060">');
htp.bold(' Nhập vào Họ tên , ngày sinh');
htp.print('</td></tr>');
htp.print('<tr><td>');
htp.print('<table width ="95%" border=0>');
htp.print('<tr><td>');
htp.print('Họ và tên :');
htp.print('</td><td>');
htp.print('<input name ="ten" type= text  size=30 >');
htp.print('</td></tr>');

htp.print('<tr><td>');
```

```
htp.print('Ngày Sinh:');
htp.print('</td><td>');
htp.print('<input name ="ngay" type= text  size=10 >');
htp.print('</td>');

htp.print('<tr></tr>');
htp.print('<tr><td><br></td><td>');
htp.print('<input type="Submit"  value="OK"> ');
htp.print('<input type="Reset"  value="Reset"> ');
htp.print('</td></tr>');
htp.print('</table>');

htp.print('</td></tr>');
htp.print('</table>');

htp.print('</center>');
htp.print('</form >');

htp.print('</body>');
htp.print('</html>');

end;

-------------------------------------
procedure insert_data(ten varchar2  default null, ngay varchar2  default null)
is

begin
htp.print('<html>');
htp.print('<body>');
insert into DU.ngay_sinh  values(ten, to_date(ngay,'dd/mm/yy'));
htp.print('<b>Đã Insert Dữ Liệu Vào Table </b>');
htp.print('</body>');
htp.print('</html>');


end;

-------------------

procedure nhap_dkx is
```

```
begin
htp.print('<html>');
htp.print('<Body>');
htp.print('<Form action="http://acernt:800/du/owa/demo1.hien_kqx">');

htp.print('<center>');

htp.nl;
htp.nl;

htp.print('<table width="50%" border=1>');
htp.print('<tr><td bgcolor="0069060">');
htp.bold(' Nhập vào các điều kiện xoá');
htp.print('</td></tr>');
htp.print('<tr><td>');
htp.print('<table width ="95%" border=0>');
htp.print('<tr><td>');
htp.print('Họ và tên:');
htp.print('</td><td>');
htp.print('<input name ="ten" type= text  size=30 >');
htp.print('</td></tr>');

htp.print('<tr><td>');
htp.print('Từ Ngày:');
htp.print('</td><td>');
htp.print('<input name ="tu_ngay" type= text  size=10 >');
htp.print('Đến Ngày:');
htp.print('<input name ="den_ngay" type= text  size=10 >');
htp.print('</td></tr>');
htp.print('<tr></tr>');

htp.print('<tr><td><br></td><td>');
htp.print('<input type="Submit"  value="OK"> ');
htp.print('<input type="Reset"  value="Reset"> ');
htp.print('</td></tr>');
htp.print('</table>');

htp.print('</td></tr>');
htp.print('</table>');

htp.print('</center>');
```

```
htp.print('</form >');

htp.print('</body>');
htp.print('</html>');

end;

-----------------
procedure  hien_kqx(ten varchar2  default null,  tu_ngay varchar2  default
null, den_ngay varchar2  default null) is

cursor_name INTEGER;
ret INTEGER;

ngay1  date;
ngay2  date;
ngay_sinh  date;
ho_ten   varchar2(30);
para    varchar2(1000):=null;
dem   integer := null;
ts       integer:=0;

begin
htp.print('<html>');
htp.print('<Body bgcolor="FFFFDC">');
htp.print('<center>');
htp.print('<table width= "60%"  border=1>');
htp.print('<tr><td>');
htp.print('<h2>Danh sách kết quả</h2>');

if tu_ngay  is not null then para:=para|| ' tab.ns
>='''||to_date(tu_ngay,'DD/MM/YY')||'''; end if;
if den_ngay is not null then para:=para||'$'|| ' tab.ns
<='''||to_date(den_ngay,'DD/MM/YY')||'''; end if;

if ten is not null then para:= para||'$'||' tab.hoten='''|| ten||'''   ; end if;

dem:=length(para);
if (SUBSTR(para,1,1)='$') then para:=SUBSTR(para,2,dem-1); end if;
if (SUBSTR(para,dem,1)='$') then para:=SUBSTR(para,1,dem-1); end if;
para:= replace(para,'$',' AND ');
```

```
if para is not null then
        para := 'delete Ngay_sinh tab  where ' || para;
end if;

if para is null then
        para:='delete ngay_sinh ';
end if;

   cursor_name := DBMS_SQL.OPEN_CURSOR;
   DBMS_SQL.PARSE(cursor_name, para, DBMS_SQL.V7);
   ret := DBMS_SQL.EXECUTE(cursor_name);
   DBMS_SQL.CLOSE_CURSOR(cursor_name);

   htp.print('</table>');
   htp.print('</body>');
   htp.print('</html>');
    end;
---------------------------------------
procedure test1  is
  cursor c is select hoten, ns from ngay_sinh;
  r c%rowtype;
  begin
  htp.print('<html>');
  htp.print('<Body bgcolor="006096">');
  htp.print('<table border=1>');
  for r in c
  loop
  htp.print('<tr><td>');
  htp.print(r.hoten );
  htp.print('</td><td>');
  htp.print(r.ns);
  htp.print('</td></tr>');
  end loop;
  htp.print('</table>');
  htp.print('</body>');
  htp.print('</html>');
  end;
end demo1;
/
```