Distributed Databases and Remote Access to a Database

Table of Contents

1 Distributed Databases ........................................................................................................ 2
2 Internet (Overview) ........................................................................................................... 4
3 Internet-Based Applications ............................................................................................. 9
   3.1 Server-Side Scripting ................................................................................................. 9
   3.2 Client-Side Scripting ............................................................................................... 12
4 PHP-Hypertext Preprocessor ............................................................................................ 16
   4.1 PHP Basics ............................................................................................................. 16
   4.2 Interface to a DBMS ............................................................................................... 20
1 Distributed Databases

Normally, a database can be seen as a big number of so-called physical pages, each page is a blocks of fixed size which serves as a unit of interaction between DBMS and a physical storage device.

Homogeneous distributed databases:
Physical pages may be distributed between a number of computers combined into a network.

Heterogeneous distributed databases:
Logical Components of a database (say, Relations) may be also distributed between a number of computers combined into a network.
Remote access to a database:
In a simplest case, a remote access to a DBMS may be provided via a network.
2 Internet (Overview)

*Internet* is the largest world-wide computer network that exists today. It is in fact a network of networks that is estimated to connect several million computers and with over 100 million individual users around the world - and it is still growing rapidly.

A notable feature of the Internet is that it brings together multiple hardware and operating system platforms from dozens of different manufacturers. Clearly, communication between these different platforms would not be possible unless they agree to some way of exchanging data. The *Internet Protocols* define such data exchange schemes, comprising two kinds of standards:

First is *TCP/IP*, which is an acronym for *Transmission Control Protocol/Internet Protocol*.

TCP/IP specifies the data transport layer of communication, which treats a data transaction between two computers as a stream of bytes referred to as a transport data unit. Simply put, data exchange between any two computers on the net is supported by TCP/IP if the data is sent in one or more transport data units.

*Internet Data Service* protocols are used by internet applications. There are a number of such protocols, each designed for some particular purpose. There are special protocols, for example, to support *distributed collaborative*
**hypermedia systems (HTTP)**, Internet News System (News) and File Transfer Systems (FTP).

**HyperText Transfer Protocol (HTTP)** is an example of an Internet Data Service protocol. It is designed to support communication between clients and a hypermedia information server.

- Clients send requests for certain services to a server.
- The server responds by sending back relevant data to the clients.

Some requests can also cause side effects in the information maintained by the server, such as addition or deletion of certain documents. HTTP basically defines the internal structure of supported requests and responses.

**The World Wide Web (WWW)** is a globally distributed collection of so-called WWW documents. These are in fact documents written in a mark-up language called a **HyperText Mark-up Language (HTML)**.

The pages residing on some particular host machine are made accessible over the net through HTTP. In other words, the WWW architecture is essentially that of multiple HTTP servers on the Internet serving WWW pages to HTML clients.

**The Uniform Resource Locator (URL)** is one of the most important Internet concepts. It may be viewed as a means of uniquely identifying resources on the net.
In HTTP, URLs identify the data to be transmitted. HTML allows for URLs to be embedded in its pages. This is the basic linking mechanism in WWW: the embedded URLs typically point to other HTML pages.

Thus the **World Wide Web (WWW)** can be seen as a distributed collection of multi-media (HTML) documents interrelated by means of computer-navigable links. The fact that HTML is the WWW de facto standard for describing how information is structured and displayed underlines its importance to the web architecture. It allows different vendors to develop **WWW browsers** that, while running on different hardware and software platforms, still display web pages in approximately the same way.

A mark-up code is simply an ASCII character sequence distinct from the text. Typically, text is bracketed by a start code and an end code, and the text thus enclosed is subject to the properties that the code describes. HTML mark-up codes are called **HTML tags** and are distinguished from text by adopting the following notation:

- A start tag is written as "<tag-X>" where tag-X is some reserved code identifier
- The corresponding end tag is written as "</tag-X>"
HTML tags may be used in combination to achieve multiple text emphasis effects:
eg.

\[
\text{<B> <I> bold and italics <U> and underlined; </U> </I> <BR> <FONT size=+2> this line is not underlined and 2 sizes larger; </FONT> and this is back to normal, unemphasised text}
\]

will display something like the following:

\[
\text{bold and italics and underlined;}
\text{but this line is not underlined and 2 sizes larger;
and this is back to normal, unemphasised text}
\]

An HTML document would not be a multimedia document if it only handles text. Other media objects are introduced as so-called inline objects. These objects exist as files that are separate from an HTML document and are included at appropriate points using special tags.

An image is included using the tag

\[
\text{<IMG src="file name" ...>}
\]
As mentioned earlier, a multimedia document becomes a hypermedia document with the addition of hypertext-style links. Links specified in HTML allows the browser to navigate to either a new point in the same document or to a different document. Links are introduced using the anchor tag:

\[<A \text{HREF=}"URL"> anchor </A>\]
3  Internet-Based Applications

Internet is based on the client-server architecture. There are two main methods for developing Internet-Based Information systems:

- Server-Side programming (scripting)
- Client-Side programming (scripting)

3.1  Server-Side Scripting

Most queries currently made to WWW servers fetch static data stored in a portion of the file system associated with the server.

The CGI interface provides a means for a client to request that an arbitrary program be executed by the server. The reason for running that program can be to produce side effects, such as updating a data base or sending e-mail to someone, but more often the program is run in order to return data directly to the client/user in the form of an HTML document generated by the program.
The **CGI interface** provides a very powerful mechanism for building so-called Internet-Based Information systems.

It should be especially noted that CGI applications may communicate to a file system and other software packages installed on the server. For example, CGI scripts may provide an internet access (i.e. interface) to a big local database, expert system, etc.

Generally, a CGI script is invoked by an HTTP request looking as follows:

```
http://[Uniform Resource Locator of the script] & [parameters]
```

Parameters are passed to a CGI application as a value of special environment variable "QUERY_STRING".

Values are assigned to environment variables by the server before the CGI program begins execution and, thus, are available to it when it begins.

For example:

```
http://coronet.iicm.edu/cgi-bin/getMail.cgi ? Name=Nick&City=Graz
```

```
QUERY_STRING="Name=Nick&City=Graz"
```

Parameters are typically sent as a result of processing a so-called HTML FORM.
It often represent a query string, such as a query to a database, depending on the function of the FORM. You can, of course, manually enter parameters directly in the URL.

for example:

<A HREF="http:coronet.iicm.edu/cgi-bin/sentMail.cgi?Name=Nick&Topic=Important">Click here to run it</A>

A form is introduced by the tag <FORM> and terminated by the inverse tag </FORM>. The attributes of the <FORM> tag includes METHOD and ACTION. For example:

```html
<FORM METHOD=GET ACTION="http://host/cgi-bin/script_name">
</FORM>
```

- METHOD specifies which technical protocol the web server will use to pass the form data to the program that processes it, and
- ACTION tells the server exactly which program that is.

A form field to request the user to enter text that is to be sent to the CGI script is introduced by the following tag:

```html
<INPUT TYPE="text" NAME= "name of CGI script parameter" SIZE="width of the input area">
```

Note that the input data is sent to the CGI script in the form

"Name of the parameter" = "Entered Value"

The CGI script processes the entered data and responds with a new HTML document
If a particular form contains *multiple elements*, the following tag is used to pass the submission of the input data to the CGI script:

```
<INPUT TYPE= "submit" NAME="parameter" VALUE="Value if pressed">
```

The button when pressed will send, in addition to any information entered in the form, the message "parameter" = "Value if pressed".

Note that there may be several of these input tags within a form. The VALUE attribute identifies which button, i.e. `<INPUT>` has been selected. When the user clicks the "submit" button, the browser collects the values of each of the input fields and sends them to the web server identified in the ACTION keyword of the FORM open tag. The web server then passes that data to the program identified in the ACTION, using the METHOD specified.

Server-side Internet Programming Languages:
- PERL
- Java Script
- Java Servlets
- PHP

3.2 Client-Side Scripting

Actually, Internet Browsers are also much more complex software systems than just an HTML interpreter as we saw it before.
Architecture of a Modern Internet browser includes a number of so-called **Virtual Machines** which are able to interpret a special imperative code known as **scripts** or **applets**.

Applets are normally small software applications, but they do not run standalone. Instead, applets comply to a set of conventions that lets them run within a Java-compatible browser on the WWW client.

Applets are embedded directly to HTML code using tags looking as follows:

```html
<applet code = "x.jar"
       width = "number of pixels" height = "number of pixels">
  <param name="a" value="b"/>
</applet>
```

Thus a WWW client can fetch an applet from a server site and run it locally to provide any kind of visual effects and/or interaction that is needed.

Whenever a browser encounters the applet tag

```html
<applet code = "x.jar"
       width = "number of pixels" height = "number of pixels">
  <param name="a" value="b"/>
</applet>
```

it is rendered as follows:
Distributed Databases and Remote Access to a database

- 1. A rectangle space defined by the width and height parameters is reserved on the screen;
- 2. A new virtual machine is activated and the reserved space is allocated for such machine to be used as a virtual display window;
- 3. The code is rendered by the virtual machine using parameters predefined by the applet tag.

![Diagram of applet rendering](image)

Scripts are just fragments of source code which are embedded directly into HTML documents. The code is interpreted directly by an internet browser. Scripts are embedded directly into HTML code using tags looking as follows:

```html
<SCRIPT>

...  

</SCRIPT>
```

Thus a WWW client does not need to additionally fetch scripts from a server.

![Diagram of script execution](image)

On the first glance the scripting technique seems to be very similar to applets discussed early.

In reality, these two methods are essentially different:
Distributed Databases and Remote Access to a database

- applets run more or less independently of an HTML document. Browser just allocates a virtual screen for an applet and let the virtual machine to control it. There is no way of accessing the document elements, or to modify them.

- client-site scripts may easily access elements of a current document to modify them (say, alter links, images, textual fragments, etc.)
4 PHP-Hypertext Preprocessor

**PHP** (recursive acronym for "**PHP: Hypertext Preprocessor**") is a widely-used Open Source general-purpose server-side scripting language that is especially suited for Web development. There are three PHP features that make it, perhaps, a most popular tool for developing information systems based on the Internet:

- embedding PHP scripts into ordinary HTML pages what allows to combine expressive power of both languages.
- flexible interface to many modern Database Management Systems (MySQL, Oracle, Sybase, mSQL, Generic ODBC, and PostgreSQL)
- possibility to dynamically output images and other multi-media files

### 4.1 PHP Basics

**PHP** is what is known as a *server-side scripting language*. Thus the language interpreter must be installed and configured on the server before one can execute commands.

Now, we assume that your Web server has the PHP support activated and that all files with the extension `.php3` are handled by PHP interpreter. If that's the case just create `.php3` files, put them somewhere in your Web server directory and the server will parse them on a request, outputting whatever the result of the execution may be back to the client. There is no need to compile anything.

So, let us start, as so many times before, with a file called `hello.php3` that will produce a simple output: "Hello, World" enclosed by some HTML tags. The code of a PHP program may look as follows:

```html
<html>
<head>
<title>PHP Test</title>
</head>
```
Distributed Databases and Remote Access to a database

```php
<? PRINT "Hello, World"; ?>
</html>
```

The PHP interpreter returns the following HTML file:

```html
<html>
<head>
<title>PHP Test</title>
</head>
<body>
<B>I say "Hello, World"
</B>
</body>
</html>
```

Alternatively, the PHP script may be embedded into HTML using tags looking as follows:

```html
<html>
<head>
<title>PHP Test</title>
</head>
<body>
<B>I say
<script language = "php">
 PRINT "Hello, World";
</script>
</B>
</body>
</html>
```

Variables in PHP are represented by a dollar sign followed by the name of the variable. The variable name is case-sensitive.

```php
<?
$a = "Nick";
$A = "Denis";
echo "$a, $A"; // outputs "Nick, Denis"
?>
```

In PHP, variable types are always assigned by types of values.

PHP control statements are almost identical to control statements in C and Java programming languages. (See, for example, "while" control statement below)

```php
<?
$i = 0; // integer
$length = 3;
$A[0] = "First"; // array of strings
?>
```
The script above would return the following HTML fragment:

```
First<br>&gt;Second<br>&gt;Third<br>&gt;
```

Consider the following HTML form:

```
<form action = "action1.php3" method = "POST">
Name: <input type = "text" name = "name" size = "20">
<br>
I prefer:
<select name = "preference" >
<option value = Movies >Movies</option>
<option value = Music >Music</option>
<option value = Theater >Theater</option>
</select>
<br>
<input type = "submit" value = "Send it!" >
</form >
```

After entering the requested info and pressing "Send it!" button:
The client will send the following HTTP request to the server:

```
http//[host]/[path]/action1.php3?name=[value]&preference=[value]
```

For example:

Name: Nick
I prefer: Theater

Would produce:

```
http//[host]/[path]/action1.php3?name=Nick&preference=Theater
```

The HTTP request:
```
http//[host]/[path]/action1.php3?name=[value]&preference=[value]
```

Is interpreted as follows:

- Server creates two environment variables:`$name` and `$preference` with values received as a part of the request.
- Server invokes the script `action1.php3` from the specified directory.
Distributed Databases and Remote Access to a database

- Variables: $name and $preference can be processed by PHP imperative statements as usual

The script will handle the variables passed from the form mentioned above:

```php
<?
echo "<center>";
echo "Hello, $name.";
echo "<br>";
echo "You like $preference.<br>";
echo "Thank you for your cooperation.";
echo "</center>";
?>
```

A function may be defined using syntax such as the following:

```php
<?
function fact ($arg)
{
    $retval = 1;
    var $i = 1;
    while ($i <= $arg)
    {
        $retval = $retval*$i;
        $i++;
    }
    return $retval;
}
$f3 = fact (3);
echo "f3"
?>
```

PHP supports rather powerful library of predefined functions. There are functions that you may use to send emails, open network connections or calculate trigonometric functions. A big family of standard PHP functions allows to manipulate with data residing on different database servers, such as MySQL server, Oracle server, etc. As a very simple example, we can call a standard PHP function called "date". This function returns the current date in a specified format:

```php
<?
$today = date("Y-m-d");
echo "<center>";
?>
```
4.2 Interface to a DBMS

Standard PHP distribution comes with a number of standard functions which allow scripts to communicate to a wide range of currently popular database management systems (DBMS). There are, for instance, function libraries for manipulating MySQL databases, Oracle databases, Informix database and others.

Normally a database transaction is carried out as the following sequence of actions:

- connect to a DBMS (there may be a DBMS installed on the same server or on another Internet Server);
- select a database (there may be a number of databases accessible via a single DBMS);
- send a query as a string to the DBMS;
- get a result as an array of tuples;
- disconnect;

Consider the following database:

- **Customer(C#,Cname,Ccity,Phone)**
- **Product(P#,Pname,Price)**
- **Transaction(C#,P#,Date,Qnt)**

Suppose, the database is supported by MySQL DBMS.
A simplest query:

"Get product names for products bought by customer number 1"

is implemented by the following script;

```php
$hostname = "localhost";
$username = "student";
$password = "student";
$dbName = "MyFirm";
MYSQL_CONNECT($hostname,$username,$password);
MYSQL_SELECT_DB("$dbName");
$query = "SELECT Pname FROM Product,Transaction";
$query = "$query WHERE C# = 1 AND";
$query = "$query Product.P# = Transaction.P#";
$result = MYSQL_QUERY($query);
?>
```

Obviously, the script can generalized to allow users to input arbitrary customer number (C#) and select products bought by the customer.

**User Interface = HTML Form**

```html
<form action = "query.php3" method = "POST">
Customer: <input type = "text" name = "cnumber" size = "3">
<input type = "submit" value = "Send it!">
</form>
```

```php
$hostname = "localhost";
$username = "student";
$password = "student";
$dbName = "MyFirm";
```
MYSQL_CONNECT($hostname,$username,$password);
MYSQL_SELECT_DB("$dbName");
$query = "SELECT Pname FROM Product,Transaction";
$query = "$query WHERE C# = $cnumber AND";
$query = "$query Product.P# = Transaction.P#";
$result = MYSQL_QUERY($query);
?>

From a programmer's point of view, the query result is a two-dimensional table where

- rows are addressed by an index
- columns are addressed by an unique name

<table>
<thead>
<tr>
<th>P#</th>
<th>Pname</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CPU</td>
<td>1000</td>
</tr>
<tr>
<td>2</td>
<td>VDU</td>
<td>1200</td>
</tr>
</tbody>
</table>

The table can be process by means of two functions:

- **MYSQL_NUMROWS** returns a total number of the table rows
- **MYSQL_RESULT** returns a value of particular table element

Thus,

- **MYSQL_NUMROWS($result)** returns "2"
- **MYSQL_RESULT($result, 0, "Pname")** returns "CPU"
- **MYSQL_RESULT($result, 1, "P#")** returns "2"
- **MYSQL_RESULT($result, 1, "Price")** returns "1200"

Generally speaking, the result should be returned to the client in a form of a correct HTML file.

<?http
...

<?http
$query = "SELECT Pname FROM Product,Transaction";
$query = "$query WHERE C# = $cnumber AND";
$query = "$query Product.P# = Transaction.P#";
$result = MYSQL_QUERY($query);
$r = MYSQL_NUMROWS($result);
$i = 0;
IF ($r == 0)
{echo "Customer $cname bought no products";
ELSE
{
    echo "Customer $cname bought the following products<UL>
    WHILE ($i < $r)
    {
        $p = MYSQL_RESULT($result, $i, "Pname");
        echo "<LI> $p"
        $i++
    }
    echo "</UL>";
}
?>

Data base relations can be updated using the same scripting paradigm

---

**User Interface = HTML Form**

```html
<form action = "update_product.php3" method = "POST">
&ltB>&ltCENTER>PRODUCT:&lt/CENTER>&lt/B>
Number: <input type = "text" name = "Pnumber" size = "3">
Name: <input type = "text" name = "Pname" size = "20">
Price: <input type = "text" name = "Price" size = "6">
<input type = "submit" value = "Send it!" >
</form>
```

```
<?http
$hostname = "localhost";
$username = "student";
$password = "student";
$dbName = "MyFirm";
MYSQL_CONNECT($hostname,$username,$password);
MYSQL_SELECT_DB("$dbName");
$query = "INSERT INTO Product";
$query = "$query VALUES('$Pnumber','$Pname', '$Prise')";
$status = MYSQL_QUERY($query);
?>
```