Applications of Advanced Software Techniques to Internet and WWW Server Development

John A. Brosnana

Institute of Technology, Tralee, Kerry, Ireland.

Follow this and additional works at: https://sword.cit.ie/allthe

Part of the Computer Sciences Commons

Recommended Citation
Applications of Advanced Software Techniques to Internet and WWW Server Development.

John Brosnan
Applications of Advanced Software Techniques to Internet and WWW Server Development

A thesis presented for the M.Sc. Degree of the National Council for Educational Awards

by

John A. Brosnan, B.E.(Electrical)

Supervisor: Deirdre Lillis

Submitted to the National Council for Educational Awards, September 1999
Applications of Advanced Software Techniques to Internet and WWW Server Development

by

John A. Brosnan

ABSTRACT

We are living in an information age. Never before has so much data been globally available and instantaneously accessible. The main reason for this revolution is the arrival of the Internet; or perhaps more accurately, the World Wide Web (WWW). This thesis discusses the development of Internet and WWW servers and the latest Web technologies including Java, CORBA and Object Databases. It presents the fundamental concepts of server development and aims to study in particular dynamic WWW servers providing multimedia content. From the viewpoint of interoperability, the CORBA multimedia standard is reviewed. The various mechanisms of data storage and retrieval for internet applications are examined and compared. The steps involved in the creation of a WWW server application using an Object Database are described in detail and performance factors for such an application are investigated.
ACKNOWLEDGEMENTS

Firstly, I would like to thank my supervisor, Deirdre Lillis, for all the help and advice which she has given me since the outset of this project. Without her support, this work could not have been completed. I also wish to thank the Head of School of Science and Computing, Mr. Seamus O' Shea, for telling me about this opportunity initially and for supporting me throughout my time at the ITT. I wish to thank the director of the Institute, Dr. Sean McBride, for the use of the research laboratory and associated facilities throughout the duration of this thesis. Finally I would like to thank my fiancee, Norma, for all her help and support over the years and for being there for me always.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>1</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>2</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>III</td>
</tr>
<tr>
<td>CHAPTER ONE</td>
<td>1</td>
</tr>
<tr>
<td>BUILDING DYNAMIC WWW SERVERS</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Origins of the Internet and WWW</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Web Terminology and Protocols</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1 URLs, Web Sites and Search Engines</td>
<td>4</td>
</tr>
<tr>
<td>1.2.2 HTTP Protocol</td>
<td>6</td>
</tr>
<tr>
<td>1.2.2.2 HTTP Operation</td>
<td>6</td>
</tr>
<tr>
<td>1.2.2.3 HTTP Evolution</td>
<td>9</td>
</tr>
<tr>
<td>1.2.3 File Transfer Protocol (FTP)</td>
<td>15</td>
</tr>
<tr>
<td>1.2.4 HTML</td>
<td>16</td>
</tr>
<tr>
<td>1.3 Introduction to Web Server Development</td>
<td>16</td>
</tr>
<tr>
<td>1.3.1 WWW Client/Server Architecture</td>
<td>17</td>
</tr>
<tr>
<td>1.3.2 The Web Browser</td>
<td>18</td>
</tr>
<tr>
<td>1.3.3 Common Gateway Interface(CGI)</td>
<td>19</td>
</tr>
<tr>
<td>1.3.4 The Web Server</td>
<td>21</td>
</tr>
<tr>
<td>1.3.4.1 Handling Client Requests</td>
<td>21</td>
</tr>
<tr>
<td>1.3.4.2 System Security and Traceability</td>
<td>22</td>
</tr>
<tr>
<td>1.3.4.3 Static and Dynamic WWW Servers</td>
<td>23</td>
</tr>
<tr>
<td>1.4 Requirements of a Good Web Site</td>
<td>25</td>
</tr>
<tr>
<td>1.4.1 Availability</td>
<td>26</td>
</tr>
<tr>
<td>1.4.2 Graphics and Multimedia</td>
<td>27</td>
</tr>
<tr>
<td>1.4.3 Style Considerations</td>
<td>28</td>
</tr>
<tr>
<td>1.5 Problems Experienced by Dynamic Web Sites</td>
<td>31</td>
</tr>
<tr>
<td>1.5.1 Server Response Times</td>
<td>31</td>
</tr>
<tr>
<td>1.5.2 Hits</td>
<td>35</td>
</tr>
<tr>
<td>1.5.3 Accessibility</td>
<td>36</td>
</tr>
<tr>
<td>1.5.4 Storage</td>
<td>37</td>
</tr>
<tr>
<td>1.5.5 Performance</td>
<td>38</td>
</tr>
<tr>
<td>1.6 Important Web Technologies</td>
<td>39</td>
</tr>
<tr>
<td>1.6.1 Java</td>
<td>40</td>
</tr>
<tr>
<td>1.6.2 ActiveX</td>
<td>41</td>
</tr>
</tbody>
</table>
# Table of Contents

1.6.3 CORBA ........................................ 43

1.7 Thesis Overview .................................. 44

CHAPTER TWO ...................................... 47

MULTIMEDIA ON THE INTERNET AND WWW .......... 47

2.1 Multimedia Evolution ................................ 48

2.2 Multimedia Types .................................. 50
  2.2.1 Bit-mapped Graphics .......................... 50
    2.2.1.1 GIF ..................................... 50
    2.2.1.2 JPEG .................................... 53
    2.2.1.3 PNG ...................................... 54
  2.2.2 Vector Graphics ................................ 56
  2.2.3 Video and Animation ........................... 58
    2.2.3.1 AVI ....................................... 58
    2.2.3.2 Animated GIF ............................... 60
    2.2.3.3 MPEG ...................................... 60
  2.2.4 Audio .......................................... 62
    2.2.4.1 WAV ....................................... 62
    2.2.4.2 AU ......................................... 63

2.3 Integrating Multimedia with the Internet/WWW .... 64
  2.3.1 Handling a Multimedia file - The Browser Perspective 64
  2.3.2 Plug-ins ...................................... 66
  2.3.3 Helper Applications .............................. 67
  2.3.4 MBONE ......................................... 70
  2.3.5 VRML and 3-D Modelling ........................ 72
  2.3.6 Java Applets ................................... 73
  2.3.7 JavaScript ..................................... 74

2.4 Problems with Multimedia on the Web .............. 76
  2.4.1 Transfer Rate and Bandwidth Bottlenecks ......... 76
  2.4.2 Size Issues .................................... 80
  2.4.3 Slow Server Response Times ..................... 82
  2.4.4 Storage and Retrieval Efficiency ................. 84

2.5 OMG CORBA Standard for Multimedia .............. 85
  2.5.1 Standard Overview ............................... 85
  2.5.2 Component Definitions ........................... 87
    2.5.2.1 Streams and Stream Endpoints ................ 87
    2.5.2.2 Multimedia Devices .......................... 88
    2.5.2.3 Virtual Multimedia Device Interface .......... 89
    2.5.2.4 StreamCtrl Interface ........................ 90
    2.5.2.5 Point to Multi-point Streams .................. 91
    2.5.2.6 StreamEndPoint Interface ..................... 91

2.6 CORBA, WWW & Distributed Multimedia .............. 92
  2.6.1 Building a Web Common Facility .................. 92
  2.6.2 Constructing a More Object-Oriented Web .......... 94
  2.6.3 Distributed Multimedia Systems .................. 96
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6.3.1 Multimedia Information Systems</td>
<td>96</td>
</tr>
<tr>
<td><strong>2.7 Web Server Application Design using Advanced Software Technologies</strong></td>
<td>100</td>
</tr>
<tr>
<td>2.7.1 Web Application Architecture</td>
<td>100</td>
</tr>
<tr>
<td>2.7.1.1 JDBC</td>
<td>101</td>
</tr>
<tr>
<td>2.7.1.2 JDBC Drivers</td>
<td>103</td>
</tr>
<tr>
<td>2.7.2 System Operation-The Client</td>
<td>107</td>
</tr>
<tr>
<td>2.7.3 System Operation - The Server</td>
<td>108</td>
</tr>
</tbody>
</table>

**CHAPTER THREE**  

**STORING MULTIMEDIA OBJECTS FOR USE ON THE WEB**  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Introduction</td>
<td>111</td>
</tr>
<tr>
<td>3.2 File System Storage</td>
<td>112</td>
</tr>
<tr>
<td>3.2.1 Background</td>
<td>112</td>
</tr>
<tr>
<td>3.2.2 Pros and Cons of the Approach</td>
<td>113</td>
</tr>
<tr>
<td>3.2.3 Sample File System Storage Application</td>
<td>115</td>
</tr>
<tr>
<td>3.3 Database Storage Mechanisms</td>
<td>117</td>
</tr>
<tr>
<td>3.3.1 Background</td>
<td>117</td>
</tr>
<tr>
<td>3.3.2 Database Management Systems - Function and Goals</td>
<td>117</td>
</tr>
<tr>
<td>3.3.3 Representation Levels in a DBMS</td>
<td>123</td>
</tr>
<tr>
<td>3.3.3.1 The Physical Level</td>
<td>124</td>
</tr>
<tr>
<td>3.3.3.2 The Conceptual Level</td>
<td>124</td>
</tr>
<tr>
<td>3.3.3.3 The External Level</td>
<td>128</td>
</tr>
<tr>
<td>3.4 Relational Databases</td>
<td>128</td>
</tr>
<tr>
<td>3.4.1 Background</td>
<td>128</td>
</tr>
<tr>
<td>3.4.2 Pros and Cons of the Approach</td>
<td>129</td>
</tr>
<tr>
<td>3.4.3 Sample Relational Database Application</td>
<td>132</td>
</tr>
<tr>
<td>3.4.3.1 Modelling the Information</td>
<td>132</td>
</tr>
<tr>
<td>3.4.3.2 Manipulating the Database using SQL</td>
<td>135</td>
</tr>
<tr>
<td>3.4.3.3 Linking the Application to the Web</td>
<td>136</td>
</tr>
<tr>
<td>3.5 Object-Oriented Databases</td>
<td>138</td>
</tr>
<tr>
<td>3.5.1 Background</td>
<td>138</td>
</tr>
<tr>
<td>3.5.2 Data Model</td>
<td>139</td>
</tr>
<tr>
<td>3.5.4 Sample Object-Oriented Database Application</td>
<td>141</td>
</tr>
<tr>
<td>3.5.4.1 Modelling the Information</td>
<td>141</td>
</tr>
<tr>
<td>3.5.4.2 Manipulating the Database</td>
<td>142</td>
</tr>
<tr>
<td>3.5.4.3 Linking the Application to the Web</td>
<td>145</td>
</tr>
<tr>
<td>3.6 Object-Relational Databases</td>
<td>145</td>
</tr>
<tr>
<td>3.6.1 Background</td>
<td>145</td>
</tr>
<tr>
<td>3.6.2 Data Model</td>
<td>146</td>
</tr>
<tr>
<td>3.6.3 Pros and Cons of the Approach</td>
<td>147</td>
</tr>
<tr>
<td>3.6.4 Sample Object-Relational Database Application</td>
<td>149</td>
</tr>
<tr>
<td>3.6.4.1 Modelling the Information</td>
<td>149</td>
</tr>
<tr>
<td>3.6.4.2 Manipulating the Database using ObjectSQL</td>
<td>151</td>
</tr>
<tr>
<td>3.6.4.3 Linking the Application to the Web</td>
<td>153</td>
</tr>
</tbody>
</table>
### Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7 Conclusions</td>
<td>156</td>
</tr>
<tr>
<td>CHAPTER FOUR</td>
<td>159</td>
</tr>
<tr>
<td>THE ANALYSIS, DESIGN AND IMPLEMENTATION OF A MULTIMEDIA OBJECT DATABASE APPLICATION FOR THE WWW</td>
<td>159</td>
</tr>
<tr>
<td>4.1 On-line Interactive Processing and Object Databases</td>
<td>160</td>
</tr>
<tr>
<td>4.2 System Analysis</td>
<td>160</td>
</tr>
<tr>
<td>4.2.1 Performing OOA on the System</td>
<td>160</td>
</tr>
<tr>
<td>4.2.1.1 System Requirements</td>
<td>160</td>
</tr>
<tr>
<td>4.3 System Implementation</td>
<td>164</td>
</tr>
<tr>
<td>4.3.1 Introduction</td>
<td>164</td>
</tr>
<tr>
<td>4.3.2 Implementation tool - Jasmine</td>
<td>165</td>
</tr>
<tr>
<td>4.3.2.1 Features of the Jasmine ODBMS</td>
<td>165</td>
</tr>
<tr>
<td>4.3.2.2 Jasmine Design and Development Facilities</td>
<td>166</td>
</tr>
<tr>
<td>4.3.2.3 Jasmine ODBMS Concepts</td>
<td>168</td>
</tr>
<tr>
<td>4.3.3 Jasmine Application Development Environment (JADE)</td>
<td>169</td>
</tr>
<tr>
<td>4.3.3.1 The JADE Development Process</td>
<td>170</td>
</tr>
<tr>
<td>4.3.3.2 The JADE Environment - Application Manager</td>
<td>171</td>
</tr>
<tr>
<td>4.3.3.3 The JADE Environment - Class Browser</td>
<td>172</td>
</tr>
<tr>
<td>4.3.3.4 Creating a New Application in JADE</td>
<td>173</td>
</tr>
<tr>
<td>4.3.3.5 Creating a New Scene in Jade</td>
<td>173</td>
</tr>
<tr>
<td>4.3.3.6 Adding Scene Items and Defining Item Behaviour</td>
<td>174</td>
</tr>
<tr>
<td>4.3.3.7 Testing, Saving, Debugging and Compiling a Scene</td>
<td>181</td>
</tr>
<tr>
<td>4.3.3.8 Creating a Distribution Executable</td>
<td>182</td>
</tr>
<tr>
<td>4.4 Conclusions</td>
<td>183</td>
</tr>
<tr>
<td>CHAPTER FIVE</td>
<td>187</td>
</tr>
<tr>
<td>RUNNING AN ODBMS APPLICATION ON THE WWW</td>
<td>187</td>
</tr>
<tr>
<td>5.1 Publishing the Jade Application on the WWW</td>
<td>188</td>
</tr>
<tr>
<td>5.1.1 Creating a “Local” Database Connection</td>
<td>188</td>
</tr>
<tr>
<td>5.1.2 Installing a WWW Service</td>
<td>189</td>
</tr>
<tr>
<td>5.1.3 Installing a JADE Web Browser Plug-in</td>
<td>190</td>
</tr>
<tr>
<td>5.1.4 Configuring System Parameters</td>
<td>190</td>
</tr>
<tr>
<td>5.1.5 Publishing the Application on the Web</td>
<td>191</td>
</tr>
<tr>
<td>5.1.6 Running the Web Application</td>
<td>192</td>
</tr>
<tr>
<td>5.2 Investigation of Database Performance Factors on the Web</td>
<td>193</td>
</tr>
<tr>
<td>5.2.1 Test Methodology</td>
<td>193</td>
</tr>
<tr>
<td>5.2.2 Analysis of Test Results</td>
<td>195</td>
</tr>
<tr>
<td>5.2.2.1 Cold vs. Warm Times</td>
<td>196</td>
</tr>
<tr>
<td>5.2.2.2 Small vs. Large Result</td>
<td>196</td>
</tr>
<tr>
<td>5.2.2.3 Morning vs. Evening Request</td>
<td>197</td>
</tr>
<tr>
<td>5.2.2.4 Local vs. Remote Request</td>
<td>198</td>
</tr>
</tbody>
</table>
### CHAPTER SIX

**CONCLUSIONS AND FUTURE DIRECTIONS**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Dynamic WWW Server Development</td>
<td>201</td>
</tr>
<tr>
<td>6.2 WWW Software Technologies</td>
<td>203</td>
</tr>
<tr>
<td>6.3 Multimedia for the WWW</td>
<td>205</td>
</tr>
<tr>
<td>6.4 Multimedia Storage on the WWW</td>
<td>206</td>
</tr>
<tr>
<td>6.5 Development of a WWW Server Application Utilising OODBMS Technology</td>
<td>209</td>
</tr>
<tr>
<td>6.6 Final Conclusions</td>
<td>211</td>
</tr>
</tbody>
</table>

### APPENDIX A

A(l)

### APPENDIX B

B(l)

### APPENDIX C

C(l)

### BIBLIOGRAPHY & REFERENCES

D(l)
CHAPTER ONE

BUILDING DYNAMIC WWW SERVERS

"I dabbled with a number of programs representing information in a brain-like way. Some of the earlier programs were too abstract and led to hopelessly undebuggable tangles."

Tim Berners-Lee, commenting on developing the World Wide Web

This chapter presents a background on the development of the Internet and how the WWW came into being. The basic concepts and protocols used in creating a Web server are discussed and dynamic Web servers are examined in detail. The problems faced by dynamic servers are illustrated and a number of Web related technologies such as Java and ActiveX are reviewed with particular emphasis on how they contribute to creating dynamic effects in Web pages. Finally the concept of CORBA is introduced and the remainder of the thesis is outlined.
1.1 Origins of the Internet and WWW

In 1969 the US Department of Defence funded a project called ARPAnet whose function was to link the department with military contractors and the numerous academic facilities engaged in military-funded research. Another goal of ARPAnet was to develop reliable computer networking technology which could withstand a military attack. A feature known as ‘dynamic rerouting’ was at the centre of the ARPAnet’s network design - if one of the links was cut or disrupted then the network would divert the message packets to other links. This meant that all computers on the network would need to be able to talk to one another and know what to do with packets of information they received. What made these tasks possible was the creation of a protocol suite known as Transmission Control Protocol/Internet Protocol (TCP/IP) [Eckel 95] which soon became the de facto protocol of network communications.

ARPAnet’s popularity quickly grew and by the early eighties it began to overload due to increased network traffic so a second network had to be created. This was christened MILNET and its purpose was to service only military sites. ARPAnet then handled only non-military network traffic but was still linked to MILNET through the software system called IP¹. Being able to handle tens of thousands of separate networks meant that IP soon became the standard and the term ‘Internet’, which originally described a system whereby inter-network communication was made possible, became the name of the global network so well known today.

¹ IP breaks up Internet messages into packets of 200 bytes, wraps each packet to facilitate easy transfer and labels its contents and destination
The ARPAnet project became the responsibility of the US National Science Foundation (NSF) in 1984 and due to increased traffic the ARPAnet was again in danger of being overloaded. NFSNET was set up to provide a faster network and worked so well that in 1990 the slower ARPAnet was discontinued. By the early 1990’s a number of large commercial companies such as IBM had created their own local networks which were linked to the Internet thus providing valuable services and information.

In 1991, the University of Minnesota created Gopher, a basic software program for retrieving textual information from servers on the Internet. This served to increase the number of Internet users worldwide but it was with the arrival of Tim Berner-Lee’s HyperText Markup Language (HTML) [Raggett 97] proposal in 1990, as a result of the global hypertext project called the World Wide Web (WWW), that things really took off. Since the WWW (or Web) program became available on the Internet in the summer of 1991 it has enjoyed unsurpassed growth to become arguably the most popular networked information system to date. All surveys on Internet hosts [Rutkowski 97] show an exponential growth curve in the time period from 1991 to July 1997 and this trend seems set to continue for the foreseeable future. It’s popularity has meant that the number of commercial web sites has grown [Gray 97] from a mere 29 in December 1993 to around 400,000 by January 1997 hence earning the Web its status as a ‘virtual marketplace’.
Chapter 1 - Building Dynamic WWW Servers

1.2 Web Terminology and Protocols

1.2.1 URLs, Web Sites and Search Engines

Every page retrieved by a WWW client from a remote Web server and displayed on their browser has a unique Uniform Resource Locator (URL), listed in the location field near the top of the browser window. The URL serves as an address for the page so it can be uniquely identified in the global network and connected to via hypertext links in other Web pages if required. An example of such an URL would be

http://brandon rtc-tralee.ie/CP400/ObjectDatabases/handout1.html

This URL specifies an HTML file called handout1.html. In the local file system its location is in the directory structure /CP400/ObjectDatabases. It resides on the machine (Web server) with the hostname ‘brandon’ in the domain ‘rtc-tralee.ie’ (ITT’s local domain). The last part of the domain sometimes refers to the country in which the machine resides - ‘.ie’ referring to Ireland in this case. The file will be accessed via the HTTP protocol in this instance.

Any large Web server typically contains hundreds or even thousands of Web pages with each page having a variable size.
A Web site contains documents having URLs which begin with a unique hostname so that

http://skelig.rtc-tralee.ie/CP100/CompScience/Assess1.html

and

http://skelig.rtc-tralee.ie/CP200/VisualBasic/Assess2.html

form part of the same site but

http://einstein.rtc-tralee.ie/CP100/CompScience/Assess1.html

is part of a different site.

Web browsers have computer programs called Search Engines which, among other things, allow the client to enter a word or group of words and the search engine goes through the entire Web and returns a list of all pages containing one or more of the given words. Some search engines need to be manually informed of new Web sites while others, called spiders [Nwana 96] or Web robots methodically plough their way through the Web daily, verifying old links and taking note of new sites. There are many different Search Engines available, each with their own individual characteristics. Normally a new Web 'surfer' will try out one or more of the search engines before eventually settling down to use just one or two. The most commonly used are Yahoo, InfoSeek, Excite, Alta Vista, Lycos and WebCrawler.
1.2.2 HTTP Protocol

1.2.2.1 HTTP Functionality

The HyperText Transport Protocol (HTTP) [Hethmon 97] is a set of rules allowing communication between Web clients and servers over the Internet. It is used to send and receive messages in the client/server system. HTML is the protocol used to create the Web pages sent as the data resource of the HTTP message. HTTP is a request-response type of protocol where the client application sends a request to the server and the server responds to the request. In earlier versions of the protocol this was achieved by making a new connection for each request. In the latest version persistent connections are introduced allowing the client and server to maintain the connection, exchanging multiple requests and responses until the connection is explicitly closed by one. Although persistent connections are possible, HTTP still remains a stateless protocol as no information is retained by the server between requests.

1.2.2.2 HTTP Operation

Three general request-response mechanisms exist in which HTTP operates.

- In the first case a user agent makes a connection to the origin server on TCP port 80 and sends its request. The server will be listening on this port for incoming connections and
start a new thread or process to serve the new request. Once the request has been processed the server sends the response back over the connection. (See Figure 1.1a)

Figure 1.1a

- The second case involves a proxy or cache agent as an intermediary. Here the user agent makes its requests to the proxy instead of the origin server. The proxy then makes the request to the origin server on behalf of the client. The server replies to the proxy which relays the response to the user agent, thereby fulfilling the request. This mode of operation is most often seen in firewall environments where the local LAN is isolated from the Internet. A variation on this mode of operation involves the intermediate agent also acting as a caching agent. When making a request the cache agent tries to serve the response from its internal cache of resources. The cache itself saves any response it receives (if the response is encachable). This operation mode shortens the request-response chain, improves response time and reduces network load. (See Figure 1.1b)
In the final scenario there is also an intermediate agent acting as a 'tunnel'. A tunnel operates by blindly tunnelling requests and responses between two HTTP applications. It is different from a proxy since the tunnel itself does nothing to the requests in terms of rewriting headers or requesting authentication. Tunnels are most often used to route HTTP traffic over a non TCP/IP link. (See Figure 1.1c)
1.2.2.3 HTTP Evolution

The first implementation of HTTP is known as HTTP/0.9 and with this version of the protocol, the client program could make a connection to the server on TCP port 80. The client could request information using the GET command. The major problem with this version of the protocol was the limitations it imposed. Only HTML text documents could be served - therefore multimedia based content could not be accessed. Also there was no method for a client to submit information to the server.

HTTP/1.0 essentially developed from the need to exchange more than simple text information. The first major change from the 0.9 specification was the use of Multipurpose Internet Mail Extension (MIME) -like headers in request messages and response messages. On the client side the request message grew from the simple one line request to a structured stable multiline request. The extra information in the requests included sending preferences for the type of information desired. This was expressed in terms of MIME media types and terms such as ‘text/html’ and ‘image/gif’ were initiated so Web clients and servers could send information each could use and understand. Now servers could send image and audio files along with the HTML document, thereby enriching the Web pages they resided on.

Another important feature of the 1.0 specification was the ‘if-modified-since’ header which allowed clients to implement conditional retrievals. This header let clients request that the resource be returned only if it had changed since the given date. This
meant that clients could cache frequently requested pages and update them only when necessary thereby saving much needed time and bandwidth.

Another important change was the definition of new request methods. Along with the GET request, HEAD and POST were added. HEAD requests allow a client application to request a resource and receive all of the information about the resource without actually receiving the resource. This has uses in applications such as Web robots and spiders. The POST method is what brought real interactivity to the Web for the first time. Now clients had a way to send substantial information to a server for processing. With POST came the use of the Web for inputting information: order forms, surveys and requests could be made from a Web page.

HTTP/1.0 also introduced the idea of restricted access to resources. A server could require a client to supply a username and password before returning certain resources. The idea of basic authentication allowed for the construction a Web site with private information. Such data could be restricted to a certain group of people. This also allowed a Web site to track a person throughout his visit. These security features meant that commerce could now flourish much easier on the Web.

During 1995 and 1996 the Internet Engineering Task Force (IETF)/HTTP Working Group worked to develop HTTP/1.1 to improve HTTP's general capabilities and fix some problems which had appeared.
HTTP/1.1 defined four new methods, the two most important ones being called PUT and DELETE. PUT allows a user agent to update or create a new resource on a server. In use an HTML editor might implement this as a way for the user to maintain pages on a Web site. DELETE allows a user agent to request that a particular URL be removed from the server. Through DELETE and PUT user agents can now create, replace and delete resources on a server. However, access to both methods should be controlled in some manner to prevent possible damage - via one of the authentication methods within HTTP for example.

The OPTIONS method is used to query a server about the capabilities of a specific resource or about the server in general. A user agent can determine using an OPTIONS request the methods the server supports when accessing the particular resource concerned.

The TRACE method is used for debugging purposes at the application level. A client program can use the method to have its original request echoed back to it. Using this information a client could then debug problems which might occur to an origin server when several intermediate agents handle its requests.

HTTP/1.1 implemented for the first time the idea of persistent connections as the default behaviour. Essentially this meant that once a client and server open a connection, the connection remains open until one or the other specifically requests that it be closed. Now clients could send multiple but separate requests and the server could respond to them in order. Clients are also free to send multiple requests without waiting for the responses -
pipelining the requests in effect. Implemented well response time to the user will be high, without the inefficiencies of individual requests.

One problem arising for servers as a result of persistent connections is that they cannot now signify the end of a response simply by closing the connection as before. The server needs to be able to determine the length of any entity it sends to the client. For most resources this is not a problem - the operating system can determine the length of HTML and image files. But problems arise for dynamically generated responses. HTTP/1.1 provides a solution to this problem in the form of 'chunked encoding'. Here a server or CGI process can send back an entity body of unknown initial length by sending it back in chunks of known length (See Figure 1.2). The server sends the size of the upcoming chunk in bytes and then the actual chunk of data. This is repeated until all the data is sent at which point a final size of zero is sent indicating the end of the data. This may optionally be followed with footers or header fields.

**Figure 1.2**

<table>
<thead>
<tr>
<th>Size</th>
<th>Chunk of Data</th>
<th>Size</th>
<th>Chunk of Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Size</th>
<th>Chunk of Data</th>
<th>0 size</th>
<th>Header /Footer</th>
</tr>
</thead>
</table>

*HTTP/1.1 Chunked Encoding Format for Dynamic Responses*
HTTP/1.1 also introduced ‘byte range operations’ which allow the user to select a certain byte range when downloading. This can often improve response time since a connection sometimes fails with most of a download completed. Now only the missing part needs to be downloaded and not the entire document.

With the ‘content negotiation’ feature the user agent is able to select the best representation of a single resource available. Three different methods of negotiation are possible but _transparent negotiation_ is the most useful form where an intermediate cache provides negotiation by examining the client’s `Accept` headers and sending the best response. This mechanism offloads server duties onto cache agents and improves response times to clients while providing accurate responses.

‘Digest Authentication’ is a new feature and a major improvement over the Basic Authentication of HTTP/1.0 since the user’s password is now a shared secret between client and server and no longer passed in clear text over the network. The server and client compute a ‘digest’ value via the Message Digest 5 (MD5) algorithm. However, Digest Authentication is only reasonably secure. It still requires an outside mode of exchanging the password between client and servers.

The HTTP/1.1 cache model allows servers a great deal of control over the caching of responses. The `Cache-Control` directive enables servers to explicitly mark something as cacheable that would not normally be such as the response from a `POST` request. The `max-age` directive allows a server to control how long a response may be cached. The
no-transform directive prevents an intermediate agent from transforming a response in any way. For example a server sending out medical images which should be stored in their original format e.g. TIFF may interact with an intermediate agent which may normally wish to transform all images to JPEG format because of space savings on disk and in bandwidth. This would result in a loss of information which would be unacceptable in this instance, hence the need for the no-transform directive here.

Table 1.1 below shows a list of the most important traits associated with each version of the HTTP protocol.

**Table 1.1**

<table>
<thead>
<tr>
<th>HTTP Version</th>
<th>Main Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>GET Requests. HTML text-only documents can be handled.</td>
</tr>
<tr>
<td>1.0</td>
<td>MIME headers accommodating multimedia information transfer. Conditional Retrievals. HEAD and POST Requests. Basic Authentication.</td>
</tr>
<tr>
<td>1.1</td>
<td>PUT, DELETE, OPTIONS and TRACE to facilitate server management. Persistent connections. Byte-range operations. Chunked Encoding Content Negotiation Digest Authentication Advanced Cache Control</td>
</tr>
</tbody>
</table>
1.2.3 File Transfer Protocol (FTP)

This is a set of rules which allows for the transfer of files from one computer to another over a network. Traditionally FTP has been used as a mode for making software available to the public domain. With this protocol there is no restriction on file formats so a postscript (.ps) file can be transferred just as easily as an HTML document. Once a user can make the connection to the FTP site using an FTP client such as CUTEFTP, he may receive resources from or place resources on the remote machine. A login and password are normally required but sometimes a user can make an anonymous ftp connection where the login is the word ‘anonymous’. It is also possible to access an FTP site directly from a Web browser. A remote node may be configured so that it has an FTP service (for storing documentation in various file formats). For instance companies, which are involved in research especially, often have documentation available in Adobe's PDF or Postscript formats for public inspection in a /pub directory. Often there will be hypertext links on a Web page referencing an FTP site. The URL location field on the browser will display an URL having a format similar to the following

ftp://UNIX.hensa.uk/pub/information.ps.

A connection to this site will automatically download, via the ftp protocol, the postscript file information.ps residing on the server UNIX.hensa in Britain (.uk) to the hard disk of the client’s machine. This author has made use of the FTP facility on numerous occasions during the course of this research using both an FTP client and Web browser.
1.2.4 HTML

HTML [NCSA 98] is the common ‘language’ of the WWW. It is a collection of platform-independent styles that define the various components of a WWW document. An HTML document is a plain-text (ASCII) file which can be created using any text editor (such as Textedit or VI on UNIX machines or notepad on Windows platforms). With such editors, however the HTML author can only imagine what the resulting page will look like when viewed using a browser. There are WYSIWYG HTML editors available now such as MicroVision’s WebExpress product which allow HTML documents to be designed visually. Instead of writing markup tags in a plain text file the author can view how the document will look as it is written.

Like HTTP, HTML has undergone many changes over the past few years and the HTML 4.0 Specification [W3C 97a] shows support for more multimedia options, scripting languages, style sheets and documents which are more accessible to disabled users. It also leans towards the internationalisation of documents in an effort “to make the Web truly World Wide”.

1.3 Introduction to Web Server Development

A Web server is essentially a computer system that stores information in a format which is compatible with Internet and WWW standards and which can be read by a Web
browser. A Web server can therefore be simultaneously part of the WWW and the Internet. This differentiates it from other machines such as dedicated FTP servers which may be part of the Internet but not necessarily the WWW.

In order to appreciate how a Web server may be developed it is essential to have an understanding of the interactions of such servers with the various components comprising the entire Web system. Knowledge of the protocols used for communication over the Web is equally important and both matters will be discussed in the following sections.

1.3.1 WWW Client/Server Architecture

The WWW owes much of its success to the simplicity with which it allows users to provide, use and reference information distributed geographically around the world. It achieves this through a system architecture called client/server computing. In this formation the client system is normally the desktop PC or workstation. The server is usually a bigger system which is capable of storing large amounts of data (normally in databases) and running major Web applications. Provided they are connected to an Internet or intranet network, a user at the client system can access web applications through a Web browser (See Figure 1.3).

---

2 An intranet or 'internal Web' applies Internet conventions and the WWW approach to the internal network of a company or organisation
1.3.2 The Web Browser

A browser is a platform-independent piece of software which potentially allows a client full multimedia access to the Web. The first graphical Web browser, called Mosaic, was developed by Marc Andressen at NCSA in 1993 and his creation prompted the beginning of the WWW explosion. Many different types of browser now exist, some still being limited to viewing just textual information (such as Lynx), while other more popular browsers like Netscape Communications’ Netscape Navigator and Microsoft’s Internet Explorer allow users to access almost any type of multimedia content such as image and audio files [Digital 96].
When a client attempts to access an information service on the WWW, their browser submits a remote request across the network using the hypertext Transport Protocol (HTTP). In reply to each request the user will receive an HTTP response with Hypertext Markup Language (HTML) formatting which the browser will display on the computer monitor. The client continues through this cycle until the user accesses some other system or simply terminates the Web browser.

1.3.3 Common Gateway Interface (CGI)

CGI 1.1 [NCSA 97] is the current version of a standard which allows external gateway applications to interface with HTTP information servers. Plain HTML documents retrieved from a server are static. However CGI programs are executed in real-time and so can output dynamic information. These programs allow the WWW to encapsulate third-party information services such as databases and allow users to query them. In essence, CGI defines the format of the data stream between the WWW server and gateway and the environment variables available to the gateway. In standard CGI, in response to an HTTP request, the Web server process forks and executes the appropriate program, which terminates after providing the response.

CGI programs reside in a special directory at the server machine at a location where the server knows it must execute a CGI program rather than just display its file contents on the client’s browser. They can be written in any scripting language such as PERL, TCL or

---

3 A gateway is a program or script which runs external programs under the direction of a WWW service
UNIX shell and alternatively in any high-level language that allows it to be executed on the system such as, C/C++, FORTRAN, VB and more recently, Java. However, most CGI developers prefer to use scripts instead of code as they are easier to debug, modify and maintain than a typical compiled program.

At present the primary mechanism of access to Server databases from a browser is through the use of HTML forms. These “fill-out” forms operate whereby the client program migrates to the WWW browser, asks the user to input parameters and returns these to the HTTP server for processing. Each form input item has an associated name tag so that when the user fills in a data value for such an item, the tag information is encoded into the form data. Hence, the form data becomes a stream of name=value pairs separated by the ‘&’ character. The user then submits the form data, it reaches the HTTP server and the server triggers the activation of an appropriate CGI program which either uses the information itself or passes it to some other process. Each name=value pair is URL encoded and so must be decoded in order to be of use for processing at the server side. Such decoders have been developed for most languages and are widely available on the Internet.

For high performance demands (more than 10 requests per second throughput) , especially for commercial applications, the Web server will usually be used in Fast CGI mode. This is called NSAPI by Netscape and ISAPI by Microsoft. These APIs can achieve significantly higher performance than CGI because an external function may be
called without creating a separate process [Netscape 95]. However the programming of these interfaces is generally considered more difficult than CGI [Nance 97].

1.3.4 The Web Server

As in the case for browsers, Web server software abounds with the most popular including Apache, CERN httpd, NCSA httpd, Microsoft’s Internet Information Server (IIS) and Netscape Communications Server. Each has its own advantages and disadvantages, certain server software supports certain features so it is important for someone setting up a Web site to know their server requirements before choosing the appropriate software to best fit their needs. Factors such as platform, server logging, server security and, of course, cost will be important in making the right choice.

1.3.4.1 Handling Client Requests

The Web server is software which runs on a machine at the server site. It accepts HTTP requests from Web clients (such as browsers), routes them as Common Gateway Interface (CGI) requests to the appropriate application servers and returns each HTTP response to the requesters. Based on the Uniform Resource Locator (URL) of the HTTP request the server can be configured to route each request to the appropriate application server. It is possible to dedicate a machine just for running Web server processes although adequate performance can often be achieved by running all Web server processes on the same
machine as the application server processes. Each web server process is capable of concurrently handling multiple client requests.

1.3.4.2 System Security and Traceability

Web servers can protect the server site by routing initial requests through an authorisation software subsystem which validates the identity of the requester and handles any exceptions which may arise. Some systems can also carry out financial transactions and even collect payments on a credit card number using these security mechanisms. Numerous Web applications exist where users are required to register in order to obtain a particular service. Once registration occurs subsequent usage only requires a login ID and password. Once a user's identity is validated, the identity can be passed to the Web as an alphanumeric "cookie" which enables the Web server and application servers to reliably know the identity of the requester.

In order to prevent their internal network system from violation from external network influences most organisations institute a firewall [Ablan 96] which is a computer application that can provide a secure connection to the Internet. It screens and filters all connections coming from the Internet to the internal network and vice versa through a security checkpoint so that external machines on the Internet cannot reach the internal network unless it passes this security barrier successfully (See Figure 1.4). To allow their system to interact with the outside world a limited and controlled set of access routes into and out of the firewall are maintained. Normally a site maintains a few public systems
outside the firewall which can be configured to route well-defined individual requests to Web servers and other systems inside the firewall.

**Figure 1.4**

![Diagram of Basic Firewall Operation]

The issue of traceability is handled by Web servers via their logging functionality. Each request, including the requester’s id, time, and result statistics may be logged to a file. This file can be continuously or periodically processed and helps the site administrator to monitor the activity and stability of the system at large.

### 1.3.4.3 Static and Dynamic WWW Servers

Many early Web sites were implemented with *static* pages of information. These pages are created using files containing HTML and always display the same information unless modified by some agent (typically a human). Static Web sites are still very common -
most people who have a *home page* on the Web listing their personal details and interests have a static site so that the information displayed rarely, if ever, changes.

In a *dynamic* Web application [BenDaniel 97] the situation is much more complex since new information continually flows into the system from users and other data sources and effect the data which users are browsing. Such a system normally consists of a data storage mechanism, data indexing and retrieval capabilities, a Web server, application servers and other components (See Figure1.5).

Application servers are custom programs which dynamically respond to requests. Each application server consists of one or more processes that can take virtually any action necessary to fulfil the request: start up of other local/remote programs, communication with existing processes (daemons), or access databases. The application server processes are where much of the action occurs in the overall system. In a scaleable system a few application server processes may be able to handle hundreds of requests per second from thousands of active users. The system may consist of only one type of server program or there may be multiple diverse programs.

In a dynamic *customisable* system users can create personalised information portfolios which include information specific to their areas of interest by specifying ‘query criteria’. Then new information flowing into the system from users and other sources is continuously filtered through these user queries. Information matching a user’s
personalised query is pre-stored in their information portfolio for rapid display when the user subsequently accesses it.

Figure 1.5

![Typical Dynamic Web Server Architecture]

**1.4 Requirements of a Good Web Site**

What comprises a good Web site is an objective matter but with the proliferation of commercial sites on the Web today the question of what makes a Web site attractive to potential customers is a very important one. Web sites have become greatly advanced since the arrival of HTTP/1.0 - apart from being able to submit information via forms on their browser, the client can also retrieve rich content from the server they are connected to. It is the extra multimedia types such as images, video and audio that have really made
a difference to the end-user. These types have effectively transformed the Web from being a static collection of Web pages to a host of interactive, dynamic information sources. The features such as additional multimedia options and style sheets introduced by the HTML 4.0 specification has ensured that future Web development will be even more interactive and available to more end-users.

1.4.1 Availability

HTML may be the universal language of the Web but different Web clients can view markup in different ways which is not always desirable. In effect HTML describes what your information is and not how it should be displayed. A general rule of thumb to ensure that an HTML document is readable and usable for a wide range of browsers is [Tilton 97]

"If you mark up a document so that the information is labelled as what it is instead of as how it should be displayed then browsers will render it in a way that is appropriate and professional looking"

Effectively this means that since there is a diversity of Web clients available, the HTML author must trust their markup since it is not feasible to anticipate how every browser will render the HTML document. Following this rule guarantees that, although you may not achieve the best possible result on one particular browser, you will get a very good rendering with all browsers.
Software programs like WebLint [Bowers 97] can be used to catch semantic errors in HTML and correct them. Styleguides can also be used which point out common errors made in HTML composition and suggest good practices to follow. These solutions ensure that, apart from a document reaching the widest possible audience, it will not look ‘ugly’ on any browser, which is the risk taken in disregarding recommendations and tweaking markup for certain browsers.

1.4.2 Graphics and Multimedia

Although still vital, the main attraction of a Web site is no longer Hypertext or an easy to use interface. It is the flashy graphics and alluring promise of multimedia that catches the users’ eye. The bottom line is that graphical images and multimedia content can and should be used - but conservatively. It is possible to make a Web document look unattractive by using too many images.

In addition, the size of multimedia content is an important consideration since this directly effects the time taken to download an HTML document. Unfortunately, a user will simply lose interest if they find it is taking a very long time to download your document (they will simply click on the STOP button of the browser in mid-download and move somewhere else in the Web). In this thesis, much debate will focus on the problems associated with multimedia size and how it may be addressed.
There is also the problem that some users will not be able to view images at all. In this case the ALT attribute should be used which allows alternate text to be specified for an inline image. This is very useful for images which have a specific meaning and provide a link to other documents since this meaning can be lost on browsers which do not load images. Also a small set of navigational icons which appear in every page of your Web can help download time [Ginsburg 97] since most browsers cache documents and images, therefore using the same icons (and the same URL to refer to them in the Web server) means the user only incurs the cost of downloading these images only once.

1.4.3 Style Considerations

The issue of standardisation is an important topic when one is dealing with creating Web pages. Different flavours of HTML are currently in practice including HTML2.0, Netscape's extensions to HTML2.0, HTML3.2 and HTML4.0. HTML2.0 is most likely the version that can be assumed "safe" for all browsers. HTML4.0 and the Netscape extensions are not widely implemented, let alone standardised but there is the fact that Netscape's extensions and some of HTML2.0 features are supported by one of the most popular Web browsers - Netscape. This means that some Web authors take the approach that since most people use Netscape, it is therefore fine to use Netscape elements even though it is to the detriment of people using other browsers. Others feel that nothing more than HTML2.0 should ever be used so any benefit which could be derived from later versions is lost. It is generally accepted that as long as two or more popular browsers
support a particular extension then it can be used with confidence. When designing a document for the Web the author should consider the effect that non-standard elements will have if not recognised by a browser. If it is not possible to use the elements on all clients, one solution may be to use multiple copies of the document and using content negotiation on the server to provide the reader with the correct version of the document.

The process of notification is another simple yet, often overlooked, feature of Web page design. This involves signing and time stamping an HTML document to assist both the Web author and user. The format of such a notification in HTML might look like:

```html
<HR>
Last Modified: February 8, 1998
<ADDRESS>
<a href="http://brandon rtc-tralee.ie/CompSc/">John Brosnan</a><br>
<a href="mailto:brosnanj@itsrralee.ie">brosnanj@itsrralee.ie</a>
</ADDRESS>
```

The date should be in the unambiguous format shown and there should be links as shown to the author's home page and e-mail address if they exist. This allows the user to obtain further information about the author or work done by him. It also allows the user to send the author feedback on the article in the Web document.

Hypertext links are another important consideration in a Web document and should be meaningful and intuitive for the end-user, describing exactly what the link is about. The choice of words used for a link's text is quite important since often the link text is used to
“bookmark” a Web page electronically and unless a suitable link name is given the user can end up not recalling the contents of a document in their bookmark list.

Any larger Web site should also provide a clear ordering of information by subject (Table of Contents) or some other form of entry into an infostructure (Searchable Index, What’s New - a time-oriented ordering for those already familiar with a site). Obviously, the user needs to be able to find what they are looking for and a good overview which allows the reader to quickly find a particular topic is invaluable. In addition, HTML documents should not be excessively lengthy and if appropriate should be decomposed into multiple files. An example would be breaking a book-length work into chapters which themselves are broken into sections, subsections and so on. Due to the length of time involved in retrieving documents, making a document available in readable chunks means the reader can use the information without being overwhelmed by loading times and correspondingly large amounts of information presented in a single, huge scrolling document. Access to the original Table of Contents or equivalent should always be provided on each Web page. This is very important when other authors create links to documents in your Web site but not specifically to your main entry points.
1.5 Problems Experienced by Dynamic Web Sites

1.5.1 Server Response Times

Research [Nielsen 97] on a wide variety of hypertext systems has indicated that users require response times of less than one second when moving from one page to another if they are to navigate freely through an information space. Research on human factors in relation to response times also shows the need for response times under a second. For example, studies at IBM in the 1970s and 1980s found that users were more productive when the time between hitting a function key and getting the requested screen was less than a second.

However, subsecond response times on the WWW will not be achievable for some time to come. For now at least it is recommended that the goal should be to supply pages to users in under 10 seconds as that is approximately the limit of a person’s ability to keep their attention focused while waiting for a response.

More recent Web user surveys [Georgia Tech 97] indicate that Web users are replacing slow modems (14.4kbps) with fast modems (28.8kbps or 33.6kbps) although the percentage of users connecting via some type of modem remains static at around 75%. Even worse is the fact that the number of users connecting at high bandwidths (1.5Mbps or better) is decreasing, even though the WWW requires at least this speed in order to
work well. Figures for mid-band usage (ISDN, leased lines and so on) show an increase but in reality mid-band speeds are insufficient for decent Web response times.

The real problem lies in the fact that the user's experienced response time is determined by the weakest link in the chain from server to browser. The factors involved are:

- The throughput of the server: this should not really be a problem since the cost of hardware is the smallest part of developing a Web site. However popular sites are sometimes taken by surprise by rapidly increasing traffic and do not upgrade their machines fast enough.

- The server's connection to the Internet: many sites try to save on their connection and put off upgrading from say a 1.5Mbps to a 45Mbps even when their current connection is saturated.

- The Internet itself: even though the net keeps getting upgraded, it still has bottlenecks, especially for cross-continent connections and for use at peak hours.

- The user's connection to the Internet: connection speeds are very low for the majority of users and will continue to be for many years to come.

- The rendering speed of the user's browser and computer: usually not a big problem, although complex tables can take significant time to lay out on low-end machines.
Each of the above can introduce its own delay in getting a Web page from a server to a user. The delays involved are cumulative meaning that fast responses are not going to be achievable simply by improving a single link in the chain. When a connection is upgraded from a 14.4K modem to a 64K ISDN line, the Web performance is typically only twice as good (and not the 5 times improvement implied by the bit rates).

The above implies that Web pages need to be designed so that speed is the overriding design criterion. To keep page size small, graphics should be kept to a minimum and multimedia effects should only be used when they truly add to the users understanding of the information.

Contrary to popular thinking, conservative use of graphics does not imply boring pages. Much can be achieved using coloured table cells and creative use of different fonts. In particular stylesheets can be used to improve page design without incurring a download penalty. It is recommended that stylesheets, if used, should be linked and not embedded since a linked stylesheet only needs to be downloaded once (assuming a consistent style for your site) whereas embedded styles add to the size of every single page.

The most important issue in response time is when the user gets to see a screenful of useful information. It matters less if it takes longer to load the full page and all its illustrations if the user can start acting on some information first. Fast initial loading can be achieved by
• Reducing complex tables by splitting the information into several tables. In particular the top table should be simple and fast to render.

• Using \texttt{ALT} attributes for images so that the user can understand what they are about before they are rendered.

• Utilising a server supporting HTTP keep-alive\textsuperscript{4} : saving the overhead of establishing a new TCP/IP connection for every hit cuts latency dramatically. The experienced response time can often reduce by half using keep-alive.

Commercial products such as ANACAPA's NetScore [Sullivan 96] are now available which can be used to determine the response time of a Web site. This product is an intelligent agent used specifically for network monitoring and gives both administrators and users information about how quickly the network and desired resources are responding to the users request. This greatly aids site administrators in identifying bottlenecks in the network.

\textsuperscript{4} Keep-alive is a feature which enables a server to send all document data to a client through a single connection. By default, when a client requests a page all entities such as icons and images are retrieved through a new separate connection.
1.5.2 Hits

The term 'hit' is often used as a Web server statistic. A hit to a Web site occurs whenever any file, including HTML documents, graphics image files, programs (CGI or otherwise), audio file or video file is accessed. Hence, a hit is not an accurate count for the number of visits a site receives. In fact, often a hit count may represent ten times the number of individual users who have actually accessed the Web server.

Obviously, a high hit rate is good news for a Web site since it means that users are viewing the site in large numbers. However, hits can be a problem for Web servers that have not been designed to accept such a high rate of network traffic. At the set up of the Web site the estimated bandwidth required must determined. For example, if the average size of a Web page is taken as 35kB and the hit rate is estimated at 1000hits/day then the daily server traffic would be around 35MB of data. In an 8 hour day this would be around 1200Bytes/sec. If other services such as e-mail, telnet and ftp are also to be offered, at least 14.4Kbps (1600Bytes/sec) of bandwidth needs to be reserved just for these. In total, therefore 2800Bytes/sec would need to be accommodated and a 28.8Kbps modem would suffice here. If only a 14.4Kbps modem was used instead, the information would still get to its destination, but would take more time especially for large information packets such as graphics.
1.5.3 Accessibility

One of the most fundamental problems of any dynamic Web server will be the issue of accessibility - or rather the lack of it. Often it can be caused by one or more of a number of situations

• the end-user might not be able to connect to any Web server in general. Their machine may be poorly configured in terms of dealing with network connections. Their communications software may be inadequate or even corrupted. Their Internet Service Provider (ISP) could have configuration problems such as routers which are not recognising specific addresses. It is often the case that the user is just typing the incorrect URL. In these cases, therefore, the server itself is actually not at fault.

• the Internet itself can often be at fault. Sometimes routers between the user’s ISP and the Web server’s ISP have failed, or are just not recognising the address correctly. Natural causes and human error can play a part here also since much of the Internet runs on some type of telephone line. If a human erroneously cuts into the right cable at the wrong moment, an entire city could lose their Internet connection. Also, information packets can sometimes just get mislaid or corrupted during transfer over the Internet. It is highly recommended therefore that every access which fails should be retried immediately.
• access problems can also be due to factors at the Web server end. The routers of the ISP for the Web server may not be routing requests correctly. There may be hardware errors or even a natural cause could be the source of the problem. The routers, hubs and other connection hardware should always be checked, especially if the router is located where there is a possibility that it came into human contact. Power failure can often be the problem source at the server end. Sometimes it can happen that when a server machine boots up, the Web server software (daemon) is not started. Sometimes the configuration of the server machine hosting the Web site can be problematic.

1.5.4 Storage

It may happen that the Web server can start to hog memory depending on the type of Web server software being utilised. In some Web servers, a new httpd daemon process is started for each user connecting to the site. This can be very costly in terms of system memory if the system has other processes running which httpd must contend with, or if the server if receiving a lot of traffic. Other server software (such as Netscape Communications) automatically starts a set number of httpd processes, thus limiting the amount of system memory being used.

Apart from problems in using up excessive free memory, there is also the possibility that the server could hog actual disk space. It is not the httpd processes that will do this but rather the results of running such processes. The process could be writing to log files or
files related to the operation of the Web site and such files may not be getting deleted or compressed and archived. Also, if the site has interactive areas where the user input is continuously being recorded, these must be monitored on a regular basis to prevent excessive disk space from being consumed.

1.5.5 Performance

For some users, server response times will be the critical factor in determining if a particular Web site is performing well. However, this is really a fallacy since oftentimes it is a problem or weakness at the user-end that makes the server appear slow to respond to their requests. Realistic server performance issues are therefore more likely to come from the fact that a server cannot respond to all requests or may be taking up excessive CPU time.

In the former case, it frequently occurs that the user is making an invalid request, perhaps trying to access a page which no longer exists on the server. Also, the Internet line to the server machine may exceed the maximum number of users it can handle simultaneously. This is unlikely to occur with a 1.5Mbps (up to 100 user requests simultaneously) or 45Mbps high-speed line, however. Finally the server itself may be receiving more requests than it can handle. CPU performance meters can be run on the system to determine the average CPU utilisation (100% meaning the machine is very busy).
If the server is taking up excessive CPU time it is often worthwhile just changing the server software or getting additional CPUs or machines to run Web sites on. Using server software that limits the number of httpd processes which can be launched is very useful in preventing excessive CPU time from being used. Also using the machine solely as a Web server is highly recommended. Limiting the use of and writing efficient CGI scripts or programs can also cut down on the CPU time a server process uses in satisfying a request.

1.6 Important Web Technologies

The WWW is an information source for millions of users worldwide. People from non-computing backgrounds have embraced the Internet and WWW in a way that was never thought possible only five years ago. The ease-of-use and user-friendly face of the Web means that it is here to stay for the long term. Technologies such as electronic mail (e-mail) and Web chat services have only added to the Web’s popularity. At present much research is being conducted on making the Web even more user-friendly. New technologies are being investigated on the WWW in an effort to improve performance and efficiency within the Web system [Davies 95]. Also work is being carried out to integrate legacy systems with the Web and to facilitate component interoperability [Lassila 96]. Distributed object technology and how it might be integrated with the WWW is another hot-spot of research activity at present [Lerner 96]. In the following sections some of the latest and more important Web technologies will be discussed.
1.6.1 Java

Although recent surveys indicate that e-mail (84%) and the Web (82%) are indispensable
technologies to all users, only experts in the field see Java [Campione 97] as being an
important Web technology. This has much to do with the fact that Java is a very recent
phenomenon, having only started out as a mere concept and some code in the summer of
1995. It now stands second only to C++ [Atkinson 93] as the most widely used
development language in the world.

It was created by JavaSoft, a business unit of Sun MicroSystems Inc., who were using a
small portable language for writing programs for embedded control devices when along
came the Internet revolution. Sun saw the huge opportunity available and dedicated a
small team of developers to the task of creating a language which came to be known as
Java.

Java is an object-oriented programming language, derived from C and C++ but arguably
superior to both. It is more simplistic, portable, robust, and has an automatic memory
management system which frees up the user to get on with the task of developing real
applications. Platform-independence is a huge advantage for Java over other its
competitors and entitles it to bear the “write once, run anywhere” banner.
Also, and perhaps more importantly from a Web perspective, Java has meant that browsing is no longer a static experience. Users can now interact with their HTML documents through Java "applets". These applets are effectively Java programs which are coded specifically to run in a Java-enabled browser such as Netscape. Where once a static graph appeared in a Web page, users can now interact with such graphs, modify graphical parameters and the graph becomes updated immediately as you view the page.

Currently Sun are aiming to get formal recognition of their Java technologies as an official international standard. Applications have been submitted to ISO concerning this and in effect Sun would become its own standard body as it relates to Java technologies. Although this would be unusual Sun believe this approach to be necessary as Java is changing so quickly. Unlike the year standard in the computer industry, Web and Internet technologies pass through a new generation every three to four months.

1.6.2 ActiveX

Developed by Microsoft and the cornerstone of Microsoft's Internet technology, ActiveX is essentially a set of integration technologies which enable software components to interoperate in a networked environment using any language. ActiveX controls, like Java applets, are used as site-enhancing objects. They are also used as aids for application development and standalone programs. These controls function just like traditional Object Linking and Embedding (OLE) controls which can reside locally on a client.
machine or may be downloaded from the Internet. Such controls can be used to handle client-side interactions with the user.

A major advantage which ActiveX has over Java applets is the fact that each time a user visits a different Web page, the controls (or component applications) downloaded are saved to the hard drive. In Java an applet needs to be downloaded each time a page is visited since applets are not cached. Also applets are unable to write to the user’s hard drive or start-up an application residing on the local machine. This, of course, also means that ActiveX controls pose an inherent security risk when downloaded as no safeguards against malicious code are provided. Microsoft bank on a system of trust where control developers sign their work using a digital signature. Thus, when a control is encountered on the Web, the user can identify the author and choose whether or not to download.

Developers can use ActiveX scripting to co-ordinate the activities of ActiveX controls on their Web pages. Currently available scripting languages are VBScript and JavaScript which employ OLE automation to communicate with ActiveX controls. The ActiveX control fires OLE events which can be handled in the ActiveX Script code in response to user interactions.

ActiveX documents are an exciting piece of technology. Using Internet Explorer 3.0 it is possible to view active documents as well as HTML pages. Also, it is possible to allow users to navigate, view and edit the contents of documents which cannot be rendered.
using straight HTML alone - all within a Web browser, thereby providing the user with a consistent interface.

1.6.3 CORBA

Developed by the Object Management Group (OMG), the Common Object Request Broker Architecture (CORBA) [OMG 93] is a standardised specification for the creation of distributed, object-oriented software systems. Interaction between such components of a networked system at a high level poses many challenges due to the underlying diversity inherent in most networks. The purpose of distributed technology is to provide a communications infrastructure which abstracts low-level communication layers. Object-oriented distributed technology adds a framework for encapsulation and re-use which speeds application development and improves the potential for integration of diverse applications.

The Object Request Broker (ORB) is a software layer which allows developers to define objects which can be accessed across a network through clearly defined, high-level interfaces. Effectively, it establishes the client-server relationships between objects. Through the ORB, a client can transparently invoke a method on a server object, which may be on the same or different machine on the network. The ORB is responsible for finding an object that can implement the request, passing it a set of parameters, invoking its method and returning the results (if any). The client does not need to be aware of the object’s location in the network, its programming language, operating system or any other
system aspects which are not part of an object's interface. Through this mechanism, the ORB provides interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems.

The standardised nature of the CORBA specification has important advantages. Many different ORB implementations are available including Iona Technologies' Orbix and SunSoft's NEO - this could pose a problem if different ORBs were unable to communicate with each other. However, CORBA specifies that all (CORBA) compliant ORBs must support an ORB standard communications protocol called the Internet InterORB Protocol (IIOP) [OMG 95]. This protocol ensures that different ORB implementations can freely interoperate. An application is presented in the next chapter of the thesis which integrates CORBA with the WWW.

1.7 Thesis Overview

Chapter Two - Multimedia on the Internet and WWW

This chapter overviews how multimedia has evolved and why it has become an integral part of the Internet and WWW. The most important multimedia types from a Web aspect are discussed and their integration with the Web is outlined. Difficulties associated with the provision of multimedia content are examined and the OMG CORBA standard for
multimedia is reviewed. The integration of CORBA with the WWW is investigated and an application developed using the latest Internet technologies is presented and analysed.

Chapter Three - Storing Multimedia Objects for Use on the Web

This chapter presents the various modes by which information can be stored for use on the WWW, with particular focus on multimedia storage. The pros and cons of each approach are examined and an application of each of approach in a dynamic WWW server is illustrated. Conclusions are drawn with respect to the appropriateness of the various storage mechanisms for multimedia information.

Chapter Four - The Analysis, Design and Implementation of a Multimedia Object Database Application for the WWW

A stand-alone multimedia-based educational system is developed using an object database as the storage mechanism. Initially, the object-oriented concepts pertaining to an ODBMS are discussed. Then some of the steps involved in the object-oriented analysis and design of the system being developed are illustrated. Finally the system is implemented using the Jasmine ODBMS and a distribution executable created.
Chapter Five - Running an ODBMS Application on the WWW

This chapter discusses the various steps involved in integrating the ODBMS application, created in the previous chapter, with the WWW. The database is then tested to determine access times for various system components over certain sets of network and temporal conditions.

Chapter Six - Conclusions and Future Directions

This chapter summarises the conclusions drawn from the work done on this thesis and the directions that future research in this area is likely to take.
CHAPTER TWO

MULTIMEDIA ON THE INTERNET AND WWW

"Anybody who thinks a little 9,000 line program that's distributed free and can be cloned by anyone is going to effect anything we do at Microsoft has his head screwed on wrong."

Bill Gates, regarding Java shortly before Microsoft licensed it and cancelled their so-called Blackbird project, a would-be replacement for HTML

This chapter introduces the concept of multimedia and how it relates to the Internet and WWW. Some of the more important multimedia types are discussed and the integration of multimedia with the Web is examined. The problems associated with multimedia content are illustrated and proposals as to how these difficulties may be overcome are put forward. The integration of CORBA with the WWW is analysed along with an examination of some research on distributed multimedia. Finally, a Web application created involving HTML (including multimedia types), Java, CORBA and an Oracle relational database is outlined.
2.1 Multimedia Evolution

Similar to phrases such as ‘internet’ and ‘Web’, the term ‘multimedia’ is a relatively recent arrival to the English vocabulary; so recent in fact that many dictionaries do not yet carry a definition for it. The National Multimedia Association of America (NMAA) define it as

"A computer related process that improves the transfer of information by involving the participant's simultaneous use of two or more senses"

Nowadays when people speak about multimedia they are almost invariably talking about computer related processes. However, the concept of multimedia has been around for some time - just in different formats. [MamboMedia 97]

The first form of multimedia were newspapers and memos which utilised text and images for communication. Subsequent to this the major development was radio whose mode of message transfer was audio. Next in the evolutionary cycle came television which integrated text, images and audio to convey and control the message being sent. More recently, the computer medium has allowed a transformation into realms which are beyond the capabilities of newspapers, radios or video cameras. Computer-based multimedia allows the user to go to places which could not be accessed previously. Most importantly, it is interactive so the viewer becomes part of the presentation instead of
merely being a passive observer and hence achieves a new perspective and level of understanding not possible with other media types.

In terms of the WWW, multimedia now plays a vital part of the proceedings and most Web sites contain some form of multimedia content, as distinct from plain hypertext. Multimedia on the Web deals with data such as graphics, schematics, drawings, photos, animations, simulations, audio and video. Linking to different sections of multimedia on the current or a different document can be achieved via hypermedia links (see Figure 2.1).

Figure 2.1

A Web page illustrating hypertext links and multimedia information
2.2 Multimedia Types

In broad terms there are just three multimedia types: graphics, audio and video. Although these types exist in numerous formats only a small fraction of them are ever used on the WWW. This section focuses only on the most commonly found formats on the Web.

2.2.1 Bit-mapped Graphics

Bit-mapped graphics [Cybulski 97] refers to hardware and software that represent a graphics image as a bit-map, which is essentially a collection of pixels\(^5\) that describes the image and gets drawn onto the monitor (See Figure 2.2). Each bitmap has two important parameters associated with it:

- Bit Depth - referring to the number of bits used to store information about each pixel. Basically, the higher the bit depth, the more colours are stored in an image. The lowest bit depth corresponds to 1-bit graphics which are only capable of showing two colours, black and white (monochrome). The formula used to determine the number of colours which a graphics image can display is:

\[
\text{number of colours available} = 2^{\text{number of colour bits}}
\]

so, for example, with 16-bit colour, 65536 different colours (or shades of gray) are available.

\(^5\) A pixel (or picture element) is the most basic component of any computer graphic. It refers to the smallest item which can be drawn on a computer screen.
• Resolution - referring to the number of pixels per inch (or other unit of measure) on a monitor or printer. This parameter determines the clarity and sharpness of an image. It is usually measured in pixels per inch or dots per inch (dpi).

**Figure 2.2**

*Bit-mapped Graphic Images illustrating the concept of pixels*

Bit-mapped graphics are created using **painting** programs such as Corel’s Photo-Paint 7 Plus, Adobe’s PhotoShop and Microsoft’s Paint program. Each paint application has its own unique features but almost all allow the graphic designer to work with a set of icons, each of which has its own drawing function and these programs also facilitate the drawing of common shapes such as rectangles and circles via an appropriate icon click (see Figure 2.3).
Literally hundreds of different bit-mapped graphic image file formats exist [CICA 94], each having its own set of attributes. Some of the more popular ones include BMP, PCX and the Tagged Image File Format (TIFF). Others such as GIF, JPEG and, more recently, PNG have established themselves as the most commonly used formats on the WWW and will be examined more closely in the following subsections.
2.2.1.1 GIF

Graphic Interchange Format (GIF) is a standard which defines a mechanism for the storage and transmission of raster-based\(^6\) graphics information [CompuServe 87]. It allows high-quality, high-resolution graphics to be displayed on a variety of graphics hardware and is intended as an exchange and display mechanism for still graphics images.

The bit depth for GIF can be a maximum of 8-bits/pixel corresponding to a possible maximum of 256 different colours. This matches inexpensive computer displays very well as most PCs cannot display more than 256 colours at once and is a major reason for the fact that GIF is currently the most popular graphic file format found on the Web.

Other advantages of the GIF file format include

- It excels for images with a certain small set of colours such as those created using line art (icons for example)

- GIF files can be indexed to colour maps (palettes) meaning that they can display consistently on 256 colour systems

---

\(^6\) Raster data refers to the format of the image defined as the series of pixel colour index values making up the image. A raster is a horizontal row of pixels representing one line of an image.
• There is at most 8 bits/pixel meaning that GIF files can be very small

• Large GIF images can be *interlaced*, meaning that in the Web browser the image undergoes a four-pass display process upon decoding which outputs image lines that are spaced apart. This allows the viewer to obtain a good (albeit blurry) overall idea of what the image looks like quickly and can then decide whether to wait for the entire image to download.

• the *transparent* GIF [Bernstein 97], which has a certain bit set on one of its colour map entries so that a Web browser’s background will show through the image’s background, allows an image to blend in naturally to the browser’s background rather than appearing like it is a rectangular or square stand-alone entity on the Web page.

### 2.2.1.2 JPEG

JPEG [ISO 94] is a standardised image compression mechanism. It stands for Joint Photographic Experts Group, the name of the committee that wrote the standard initially.

It is designed for compressing either full-colour(24-bits/pixel) or gray-scale still images of natural, real-world scenes as well as continuous tone\(^7\) images rendered in 3-D programs.

Some of JPEG’s advantages include

---

\(^7\) Continuous tone refers to images that have a virtually unlimited range of colours or shades of gray
• It works well on photographs and naturalistic artwork

• JPEG compression is 'lossy'\(^8\) so that the decompressed image is not quite the same as the original, allowing JPEG to achieve much greater compression than is possible with lossless methods such as GIF.

• The actual degree of lossyness can be varied by adjusting compression parameters so the image maker can trade off file size against the output image quality. This means that very small image files are possible provided that quality is not an important issue.

• JPEG images normally take longer to decode and view than GIF but decoders can trade off decoding speed against image quality using fast (albeit inaccurate) approximations to the required calculations - in this way remarkable speed up can be obtained and this feature can be invaluable when creating Web sites involving many JPEG images.

• Similar to interlaced GIF, the *progressive* JPEG is now available which allows the browser to display a general picture of the graphic before fully downloading it, permitting the user to make an early decision on whether to stay on the Web page to view the image or move elsewhere.

\(^8\) All formats involving compression inherently involve loss of information, including GIF, but with lossy compression the losses accumulate when compression and decompression is repeated.
2.2.1.3 PNG

The Portable Network Graphic (PNG) is a file format invented specifically for on-line graphics. Although GIF and JPEG are currently the graphic formats most widely supported by Web browsers, the WWW Consortium (W3C) has recently made a formal endorsement of PNG [W3C 97], indicating that the two most popular Web browsers, Netscape and Microsoft's Internet Explorer will support it as an inline file format presently (it is now supported by Internet Explorer 4.0 and Netscape Navigator 4.0).

Similar to GIF, PNG uses lossless compression so no image detail is lost. Unlike GIF or JPEG however, PNG may be stored at many different bit-depths (8-, 24- or 32-bit) using different storage methods. Figure 2.4 illustrates a PNG image within the Netscape browser. This is an 8-bit image which requires 90kB but still requires less space than an equivalent GIF image which takes 94kB. For photographic images like this, however, JPEG remains the best route since a reasonably high-quality image can be obtained for just 15kB via a compression ration of 42:1.

A major advantage that PNG has over JPEG is that it can store what is called gamma correction information, meaning that the format has the ability to display consistently on different platforms without losing contrast or brightness in the translation. This is something the JPEG format cannot do.
In addition, PNG supports 8-bit transparency, meaning that a large amount of memory is saved compared with JPEG where 24-bit transparency is mandatory.

**Figure 2.4**

*An 8-bit PNG image displayed in the Netscape Navigator Browser*

The PNG format has not been widely adopted yet and so it remains to be seen if it will become the predominant format for still Web images in the future. Table 2.1 below shows a comparison of the three graphical file formats discussed from a number of perspectives.
Table 2.1

<table>
<thead>
<tr>
<th>Format</th>
<th>Potentially Lossless</th>
<th>Potentially Lossy</th>
<th>Possible Bit-depths</th>
<th>Interlacing</th>
<th>Transparency</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIF</td>
<td>✓</td>
<td>X</td>
<td>8-bit/pixel or lower</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>JPEG</td>
<td>✓</td>
<td>✓</td>
<td>24-bit/pixel only</td>
<td>✓</td>
<td>X</td>
</tr>
</tbody>
</table>
| PNG    | ✓                   | X                 | 8-bit/pixel
            24-bit/pixel
            32-bit/pixel | ✓           | ✓            |

2.2.2 Vector Graphics

Vector (or Object-Oriented) Graphics refers to software and hardware which utilises geometrical formulae to represent images [Valentine 97]. Such images may be created and manipulated using a drawing program (such as CorelDraw 7, Microsoft Draw and AutoCAD).

The major advantage which this technique has over painting programs is that once the object has been created it can be edited at any later stage so, for example its size or location can be altered or it may be stretched. Also, vector graphic files are normally much smaller in size than bit-mapped graphics. Unlike bit-mapped graphics, most vector graphics are resolution-independent meaning that they can be enlarged to any size without -58-
losing any detail (see Figure 2.5) whereas bit-mapped graphics begin to look 'pixelated' upon enlargement (see Figure 2.2) and they look better on devices with higher resolution whereas bit-mapped graphics always appear the same regardless of a device's resolution.

Some of the more popular vector graphics file formats include Computer Graphics Metafile (CGM), PICT and Windows Metafile Format (WMF).

**Figure 2.5**

*Vector Graphic Images Illustrating Resolution Independence*

### 2.2.3 Video and Animation

Similar to still images, numerous video file formats exist. Again, only the most important ones from a Web perspective will be discussed in this thesis.
2.2.3.1 AVI

The Audio Video Interleave (AVI) file format, defined by Microsoft, is the most common format for audio/video data on the PC and can be played on Windows, Unix and Macintosh platforms [McGowan 98]. The audio tracks in AVI files are simply waveform (WAV) audio data used by the WAVE system. Video for Windows, a helper application that handles AVI, just parses the files, extracts the WAV data and pipes it to the WAVE system. Video for Windows handles the video track itself, through its MediaPlayer application. The data in AVI is ordered in time sequentially as it appears in the file. The player application displays the video frames at the frame rate indicated in the file header and the audio at the audio sample rate indicated. The video 'chunk' should be *interleaved* (hence the file format name) with an audio 'chunk' containing the audio associated with that video frame to accomplish pairs of video and audio chunks. The video information in an AVI can be compressed in a variety of ways. Video for Windows comes with several compressors such as CinePak. The advent of the WWW and internet has created a new wave of audio and video codecs (*compressors/decompressors*) which attempt to apply advanced technologies such as sophisticated motion estimation and compression, wavelets, fractals and other techniques to achieve extremely low bit rates for the internet, essential for the limited bandwidth available and the fact that most users have only a 28.8kbps connection to their ISP.
2.2.3.2 Animated GIF

This is a type of GIF image which can be ‘animated’ by combining several images into a single GIF file [Edwards 98]. Applications which support the animated GIF standard (GIF89A) cycle through each image, frame by frame. GIF animation is not as sophisticated as other animation formats but has become extremely popular owing to the fact that it is supported by almost all Web browsers. In addition they tend to be smaller in size than other animation files such as Java applets.

Animated GIFs can be created using tools such as Adobe Illustrator, Adobe PhotoShop and the GIFBuilder shareware program. Several Web sites are dedicated to supplying animated GIFs as freeware to Web developers.

2.2.3.3 MPEG

Perhaps, more than any other technology, MPEG encompasses people’s idea of what multimedia is - moving pictures with synchronised sound played by a computer. The Moving Pictures Expert Group (MPEG) is a collection of people that meet under ISO to generate standards for digital video and audio compression. MPEG defined the characteristics of a compressed bitstream which can represent moving pictures and audio.

MPEG generally provides better quality video than competing formats such as Video for Windows and QuickTime and the format achieves a high compression rate by storing
only the changes from one frame to another instead of each entire frame. This is a type of lossy compression since some data is removed as a result of the compression process. However this loss of information is generally imperceptible to the human eye.

At present, two major MPEG standards exist, namely MPEG-1 [ISO 92] and MPEG-2 [ISO 94]. Both standards are structured into four main sections describing how the system handles synchronisation and multiplexing of video and audio data along with their compression and decoding.

The most common implementations of MPEG-1 provide a video resolution of 352x240 at 30 frames per second (fps) which gives quality slightly below that of conventional VCR videos. The newer MPEG-2 standard offers higher resolutions of 720x480 and 1280x720 at 60fps, with full CD-quality audio. This is sufficient for all the major TV standards (including the American National Television Standards Committee).

2.2.4 Audio

Most Web browsers have the ability to transfer and play sounds (with the aid of other applications). However, a multitude of digital audio formats exist since no single standard has yet been agreed upon. This thesis will only discuss those formats which prevail specifically on the Web and internet.
2.2.4.1 WAV

The Wave (WAV) format is the most commonly used and supported audio file format on the Windows platform and therefore very common on the Web [Williams 96]. Officially it only uses sampling rates of 11.025kHz, 22.050kHz and 44.1kHz but any sampling rate can be used provided the hardware supports it. For example, the MicroSound I/O Module (created by Micro Technology) provides 15 sampling rates and their MicroEditor software may be used to record or save audio samples into WAV files as 8 or 16 bits/sample [Micro Technology 97].

2.2.4.2 AU

This is the standard audio format used on SUN MicroSystem computers and it has been described as the unofficial internet standard for audio files owing to the fact that it is the most common sound format found on the Web [Perlman 95]. This is because it plays on the widest number of platforms, including Macintosh and Windows.

Even if an audio sample is recorded with 16-bit resolution, it becomes converted to 8-bit when the clip is saved in AU format which ensures that AU files are relatively small in size. NAPLAYER, which ships with Netscape, is a piece of software which can play the AU format.
2.3 Integrating Multimedia with the Internet/WWW

At present most Web browsers have the in-built capability to allow the user to automatically download and view the standard image formats such as GIF and JPEG. However, this is not always sufficient as many Web sites include information which comes in a variety of formats such as audio and video clips. The browser cannot always deal with such formats independently and may need to call on another application specifically designed to handle the format. These applications may be either plug-ins or helper applications. When a Web page is encountered which utilises a plug-in or helper application that has not been installed, the browser cannot access all of the features offered at the site and the user will typically be prompted to find and download the appropriate software.

2.3.1 Handling a Multimedia file - The Browser Perspective

When a browser such as Netscape or Internet Explorer sees a link to multimedia content in an HTML document the following occurs:

- The MIME Content-Type header field of an internet text message is examined. This informs the client of the type of data contained in the message as well as subtype information.
e.g. a Content-Type of text/plain indicates a general type of text and subtype of plain and thus the client knows the message contains text of a specific format - plain in this case. Other general Content-Types for multimedia include audio, image, video and an extra one called application which accounts for data of type different from the other four basic content types. As expected, subtypes for audio include au and x-wav and among those for the image type are jpeg and gif. postscript is an example of a subtype of the application content type.

• The Content-Type value is the key to determining the file type - it indicates what sort of data is contained in the file e.g. whether it is a plain text or Adobe PostScript file, a GIF or a JPEG file.

• A mime.types file exists which is used to match file suffixes with MIME types. This file can be added to so as to incorporate new MIME types if required. Some of the mime.types file shipped with the NCSA server is shown in Table 2.2 below.

<table>
<thead>
<tr>
<th>MIME Type</th>
<th>File Suffix</th>
<th>MIME Type</th>
<th>File Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>audio/basic</td>
<td>au snd</td>
<td>video/mpeg</td>
<td>mpeg mpg mpe</td>
</tr>
<tr>
<td>audio/x-wav</td>
<td>wav</td>
<td>video/quicktime</td>
<td>qt mov</td>
</tr>
<tr>
<td>image/gif</td>
<td>gif</td>
<td>application/postscript</td>
<td>ai eps ps</td>
</tr>
<tr>
<td>image/jpeg</td>
<td>jpeg</td>
<td>application/pdf</td>
<td>pdf</td>
</tr>
<tr>
<td>text/plain</td>
<td>txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>text/html</td>
<td>html htm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• Once the suffix and MIME type have been matched up the required helper application or plug-in is invoked automatically

2.3.2 Plug-ins

A plug-in is a piece of software that is designed to operate within Netscape (or any other compatible Web browser) in order to extend its capabilities [Williams 96]. Once a plug-in has been installed correctly for use with the browser, no other configuration is necessary and the browser can then use the plug-in just like a built-in capability. This means that when a Web page is accessed which utilises a particular plug-in, the application launches behind the scenes and the file is executed within the browser window. It appears to the uninitiated, therefore, that the browser itself is executing the file and not the plug-in.

Currently, plug-ins are almost exclusively designed only for use on the Macintosh and Windows platforms so UNIX users are deprived of most plug-in related capabilities. However, most plug-ins work equally well with later versions of both Netscape and Internet Explorer. Currently, some of the more popular plug-ins include

• QuickTime - a movie player created by Apple Computers. This works by compressing video so that it can be stored and played back on the Macintosh and Windows 95 platforms which use Netscape version 3.x. It may be downloaded from Netscape’s plug-in site.
• Beatnik - this manages the playback of Rich Music Format (RMF) and other music and sound files such as MOD, WAV, AIFF and AU. The plug-in has hi-fi sound quality comparable to high-end soundcards and sounds the same across multiple platforms. It allows a Web page to play music, not only upon opening the page, but also on an event such as a mouse click.

• Weblink - a plug-in enabling Adobe’s Acrobat Exchange users to create links from Portable Document Format (PDF) documents to documents on Web servers. URLs may be embedded in PDF files similar to HTML tags. Support is available for Netscape and Mosaic browsers.

• WebFX - One of the newest developments in Web technology is interactive 3-D graphics. WebFX enables the user to view 3-D objects from any angle by permitting the viewer’s perspective to “walk” around the image. In addition the objects themselves can be manipulated.

2.3.3 Helper Applications

A helper application or simply ‘helper’ is an external, stand-alone software program which possesses capabilities lacking in a Web browser [BMRC 98]. In contrast to plug-ins, helpers are programs a browser can use but which were not specifically written for the browser. For example, when viewing a movie using a QuickTime helper application,
control passes from the browser to the QuickTime player which opens up its own window and plays the video. When it is completed, control is passed back to the browser.

The browser uses a file’s MIME type (the standard describing a file’s format) to determine whether it can read a file by itself or whether a helper is required. Similar to plug-ins, a helper needs to be downloaded and installed correctly but as these programs are external to the browser the user must also configure the system in order to indicate to the browser where a specific helper resides.

Configuring the system varies with the different platform and browser types. In Netscape, for example, in the browser window there is a GENERAL PREFERENCES option under the OPTIONS menu which can be used for configuring which helper application to use to view each particular MIME type. The version for Internet Explorer is very similar. UNIX workstations are configured differently and use the .mime.types and .mailto files. For example, in order to view Adobe PDF files, the .mailto file should contain the line

```
applications/pdf; acroread %s
```

where applications/pdf refers to the MIME type assigned to the PDF document and acroread refers to the external viewer program (helper application), Adobe Acrobat Reader, being used.
Unlike plug-ins, helper applications exist for most platforms, including UNIX and indeed many exist for all three main platforms. Some of the more popular helper applications are

- **XAnim** - this animation viewer for the UNIX platform supports many types of animation formats including Quicktime, MPEG, MOV and AVI.

- **Ghostscript** - this application, available for Windows and UNIX, interprets Adobe’s PostScript document format which is extensively used on the WWW.

- **xview** - the X Windows System-based Visual/Integrated Environment for Workstations. This supports interactive, graphics based applications in UNIX and is appropriate for viewing and converting, among others, TIFF files. The equivalent Windows based helper is Iview, which is very popular and is free on the internet.

- **RealAudio** - a helper which provides a high-quality, real-time audio system to Macintosh and Windows users over connections of 14.4kbps or faster. The sound is presently of AM quality and its creator, Progressive Networks, is currently working on improving this to FM (see Figure 2.6).
2.3.4 MBONE

The IP Multicast\(^9\) Backbone (MBONE) is a method for providing audio and video in real time via the class-D IP addressing scheme [Schulzrinne 97]. Class-D addresses range from 224.0.0.0 to 239.255.255.255 and the MBone based audio video conferences have been allocated the address range of 224.2.*.* by the Internet Assigned Numbers Authority (IANA). Therefore the MBone is a virtual network layered on top of portions of the physical internet whose special routing facilitates distributed applications in achieving

\(^9\) Network connections are essentially unicast - one machine establishes a connection with another machine and they send data and acknowledgements back and forth
time-critical "real-time" communications over wide area IP networks through a lightweight model of communication.

The IP multicast routers (mrouters) have the responsibility of distributing the multicast data stream to their destinations as opposed to individual IP hosts. The Mbone topology of mrouters is designed in such a way that it ensures efficient distribution of information packets without congesting any particular node.

MBone has been frequently used for broadcasting conferences and concerts to a limited user community, mainly scientists, for some time. The reason that the technology has not been available to a wider audience is because of the high network bandwidth (1.5Mbps or more) required for the simultaneous transfer of audio and video information. Also powerful UNIX machines, having extensions to the operating system for multicast, are required for conferences involving video, audio and a whiteboard - an area on the screen where participants can scribble notes.

It is also possible to set up telephone conference calls using the audio-only portion of the MBone, called mmphone. This sort of interaction is more straightforward than video conferencing since the required bandwidth for successful transmission is much lower (typically 64 kbps is required to carry this compared with 128 kbps for video).

Although it has been used primarily as an experimental prototype, the MBone has been used to broadcast NASA space events and many radio stations are on-line. It is not
accessible via a Web browser at present but can be accessed with a tool called CU-SeeMe on the Macintosh or Windows platform and a program called vat on UNIX machines.

2.3.5 VRML and 3-D Modelling

The Virtual Reality Markup Language (VRML), created in 1994 by Mark Pesce, Gavin Bell and Tony Parisi, is a specification which allows developers to create and navigate through 3-D worlds on the WWW [Flohr 96]. It aims to create a Web environment requiring no training and no interface. It allows the end-user to travel through exciting virtual environments on the Web instead of looking at a comparatively boring list of text and images with hypertext links. Similar to hypertext links, it is possible to jump from one 3-D VRML 'world' to another. VRML provides platform independence and is suited for low-bandwidth network connections. It doesn't provide a great level of interactivity as a consequence of the fact that it doesn't support 'real' language features but this issue is being dealt with at present and so future versions so provide more functionality. An important fact is that VRML is not an extension to HTML and HTML browsers cannot interpret it. However, a VRML page can link to an HTML document. The VRML object information resides in an ASCII text file with a .wrl extension and when a VRML enabled browser is pointed to that page, you enter the "world" described by the file. Provided a VRML viewer such as WebFX is available and a browser is configured to use the application, VRML documents can be viewed. Developers can construct complex objects from polygons and solids but light sources, object materials and other special effects can be specified also, thereby creating a true 'virtual' object. Also, rather than a geometric object, a VRML node can contain other file formats such as JPEG or a piece of
MIDI sound data. By its very nature, the fact that VRML animated 3-D scenes are heavy computing applications means that if the end-user has insufficient computing power, navigation will be sluggish and frames will be stuttered. Use of a feature called LOD (Level of Detail) in VRML will allow the user to vary the level of detail required for optimisation purposes [Pesce 95].

2.3.6 Java Applets

A Java applet is a Java program which may be included in a HTML page within the confines of an `<APPLET>` tag. When a Java-compatible browser is used to view a Web page containing a Java applet, the applets code is transferred to the end-users system and executed by the browser. The creation of a Java applet from 'first principles' is a non-trivial process in most cases and requires a thorough knowledge of the Java programming language. Needless to say, however, there are many Web sites dedicated to providing free applets for use by site developers, including some fascinating ones at Sun Microsystem's site [Sun Microsystems 99]. The fact the applets run-time code is transferred to the client machine has often been a cause of concern for would-be end users but applets loaded over the network do have security restrictions associated with them including the fact that they cannot read or write files or start any program on the host executing them. In addition they cannot make network connections except to the host from which they arrived. At the client side, a SecurityManager object is present to determine the integrity of an applet so an exception will be thrown if an applet tries to something malicious. Although an applet cannot save state on the client machine which executes it, it can do so by forwarding the information to a server-side application which could then save state to a file on its own
host. Also, an applet can play sounds (something a Java application cannot do yet), can interact with another applet on a Web page and can be configured to keep running even when the user has left the Web page on which it exists (necessary where an applet would perform some complex calculation, for instance).

2.3.7 JavaScript

JavaScript is a source of confusion quite often as it is assumed to be part of Sun Microsystems's Java Technologies. In fact, Netscape Communications Corporation [Netscape 99] created it. JavaScript is a cross-platform, object-based scripting language for client and server applications. With JavaScript it is possible to create dynamic HTML pages that process user input and maintain persistent data using special objects, files and relational databases. Through JavaScript's LiveConnect facility it is possible for JavaScript and Java applets on the same Web page to communicate. There are two veins of JavaScript, client-side and server-side, which both share the same core language (corresponding to the standardised ECMA-262 scripting language with some additions), comprising a set of core objects. It also defines other language features such as expressions, statements and operators. Client-side JavaScript includes predefined objects only relevant to running JavaScript in a browser - this is where it differs from server-side JavaScript. This JavaScript is embedded directly in HTML pages and is interpreted by the browser (Netscape Navigator 2.0 and higher) completely at run-time. When the browser requests such a page the server sends the full content of the document, HTML and JavaScript statements over the network to the client. The client then executes the
JavaScript statements as it encounters them. Client-side JavaScript statements in an HTML page can respond to user events such as mouse clicks, form input and page navigation. In terms of server-side JavaScript, embedding of statements is also the mode used. Statements can be used to connect to numerous types of relational database, share information across users of an application, access the file system on the server, or communicate with other application through LiveConnect and Java. In contrast to client-side JavaScript, HTML pages using server-side JavaScript are compiled into bytecode executable files which are then run with a Web server that contains the JavaScript runtime engine.

In comparison with Java, JavaScript is quite similar in some ways but fundamentally different in others. It supports much of the Java expression syntax and control-flow constructs. However, in direct contrast to Java's compile-time system of classes built by declarations, JavaScript supports a runtime system based on a small number of data types representing numeric, Boolean and string values. JavaScript supports dynamic binding whereas Java supports static binding. JavaScript code is integrated with, and embedded in, HTML whereas Java applets are actually distinct from HTML itself (even though they are accessed from HTML pages).
2.4 Problems with Multimedia on the Web

2.4.1 Transfer Rate and Bandwidth Bottlenecks

As the internet continues to be overwhelmed by its own popularity, due in no small part to the WWW and its increasing multimedia content, the pipes which route the internet's data continue to overflow. Some of the reasons for this are as follows:

- the internet is experiencing enormous growth in new users
- the amount of bandwidth consumed per existing user is increasing as people use technologies such as internet telephony and video conferencing
- companies are starting to demand a certain quality of service from their ISP which usually translates to a guaranteed amount of bandwidth.

Some commentators [O'Donnell 96] suggest that paying for bandwidth will potentially become a major factor in reining the internet's explosive growth. If individuals or companies are charged on the amount of bandwidth they consume rather than on the amount of time they spend on-line, the likelihood is that they will begin to take a serious look at what types of services and applications they really need and scale things back accordingly. At present some telecommunications companies, in their role as ISP, are indicating that corporations will have to pay extra to get a guaranteed amount of bandwidth for their internet traffic. This would have a knock-on effect of significantly reducing (sometimes unnecessary) internet traffic flow.
Sending simple text-only HTML files across the internet is a relatively easy task since most text files are usually (though not always) small in size. However, when sound and, more especially, video data is to be transferred across the network things become much more complex.

For example, if a picture 300x200 pixels in size is to be transferred and the picture uses only 8-bit colour (i.e. 1 byte/pixel) then there are 60000 pixels or 60kB of information for a single frame of small video. In this case the image would only take up around 20% of a normal 640x480 pixel screen. TV pictures and video, which are made up of many frames, display around 30fps and because the human eye is relatively slow (compared with the speed of light), it sees 30 sequential still pictures as continuous motion. Therefore, just to display for a second on 20% of the screen it would take 1.8 MB.

In terms of transfer, therefore, to send this small, uncompressed one second video to a user with a 28.8kbps modem (having a nominal throughput of 3600 bytes/second) would take a little over 8 minutes. Clearly, this set-up is not going to succeed and hence the need for a mechanism which can speed things up.

An obvious solution would be to increase the speed of the user’s connection. Table 2.2 below shows a comparison between the times taken to transfer the 1 second video clip over a number of different standard connections.
Table 2.2

<table>
<thead>
<tr>
<th>Connection Type</th>
<th>Video Transfer Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.4Kbps</td>
<td>16min 40sec</td>
</tr>
<tr>
<td>28.8Kbps</td>
<td>8min 20sec</td>
</tr>
<tr>
<td>56Kbps</td>
<td>4min 17sec</td>
</tr>
<tr>
<td>64Kbps ISDN</td>
<td>3min 45sec</td>
</tr>
<tr>
<td>128Kbps ISDN</td>
<td>1min 52sec</td>
</tr>
<tr>
<td>1.5Mbps (T1)</td>
<td>0min 1.2sec</td>
</tr>
</tbody>
</table>

Here the user must make the trade off in terms of economics (is the connection within financial reach - the cost of connection being generally a function of the connection speed) and speed (can the rate of transfer be tolerated). A T1 (1.5Mbps) connection would suit everyone in the sense that the transfer rate is very high. However, a T1 connection is more expensive to operate, although high connection bandwidths are desirable, and sometimes necessary, in order to receive respectable transfer rates, this alone is not enough - it would still take 72 minutes to transfer a 1 hour video via T1 connection in the above situation.

For most internet users it is not practical to rely on very fast connections from a financial viewpoint so an alternative is needed to solve the transfer problem for large multimedia files. A mechanism which achieves this is the compression technique. Essentially a compression algorithm can take a large file and compress it to a fraction of its original size, thereby reducing the amount of data which needs to be transmitted. As discussed earlier, lossless compression is achieved only if all of the original data is preserved. Both
the GIF format for still images and ZIP format for text are lossless examples. The
compression algorithms generally work by examining the set of bytes making up the
image, video or audio file and locating bytes which repeat. If upon inspection, a certain
set of $n$ bytes repeat these can then be replaced with a single flag which indicates that the
following $n$ bytes are all the same. The more sophisticated and powerful the compression
algorithm is, the smaller the multimedia file can be made and hence the reason why
developers are constantly attempting to create new and improved versions of their
compression techniques.

For JPEG and MPEG it has already been stated that there is a certain amount of
information lost in the compression but this allows much greater compression to be
achieved which is especially important if one is dealing with 24-bit colour movies, which
can be very large indeed. In compressing such files, the user has a certain amount of
control over the proceedings and can make trade-offs between what is acceptable in terms
of picture (or audio) quality and file size.

Currently much debate centres on the bandwidth problem [Maclachlan 97]. Many experts
in the area believe that the connections themselves are sufficient for most purposes and
that the servers are to blame. Recently, servers have been getting more specialised with
some content companies, for example, providing on-line video feeds to companies via
high-performance video-only servers. In an effort to save on disk space, many advertisers
are now increasingly looking toward 3-D computing as a memory cheap way to send
complex content. 3-D graphics based on Virtual Reality Modelling Language (VRML)
2.0 use a fraction of the memory of other mediums as they describe images in simple mathematical terms. It has also been suggested that internet traffic distribution models be changed to alleviate the congestion problems. Much internet traffic consists of millions of people accessing the same content. If this content was distributed in more locations, for example by local ISPs downloading the most popular content on their own servers each day, the total amount of traffic travelling long distances would significantly decrease.

In an attempt to make video and graphics more viable on the internet one company, Narrative Communications, will release in Autumn a product with which they hope to address some of the internet’s bandwidth limitations [Stahl 98]. Their server hopes to deliver multimedia content - primarily animation and graphics - in continuous streams, eliminating the need for users to download content before viewing it. The server does not require a high-speed connection such as ISDN and promises to “deliver multimedia at speeds that the average business user or consumer has”.

2.4.2 Size Issues

The amount of storage which a file takes up on a server or client machine is an important consideration, especially when those files are multimedia related. With digital video, for example, a new frame is required every 1/30th second for NTSC. Assuming 24 bits/pixel in the digital video and 30fps, the amount of disk space required for such a stream of full-motion video is shown in table 2.3 below:
Table 2.3

<table>
<thead>
<tr>
<th>TIME</th>
<th>FRAME SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>640x480</td>
</tr>
<tr>
<td>1 second</td>
<td>27.65MB</td>
</tr>
<tr>
<td>1 minute</td>
<td>1.65GB</td>
</tr>
<tr>
<td>1 hour</td>
<td>99.53GB</td>
</tr>
<tr>
<td>1000 hours</td>
<td>99.53TB</td>
</tr>
</tbody>
</table>

The table shows that a single hour of video with a resolution of 640x480 would consume almost 100GB of disk space, a figure considerably larger than most storage devices. By comparison, an equivalent amount of analogue video, having a higher resolution and an audio portion also, would only take up a quarter of a standard 240 minute cassette. Although there are devices which can store this amount of data, currently no digital storage devices exist which can store 100GB on a quarter of a device which is the size of a conventional video cassette.

Compression techniques are of paramount importance in maintaining reasonably sized data streams. Table 2.4 below estimates\(^\text{10}\) the amount of storage space utilised when compressing video of size 640x480 via two common standards, JPEG and MPEG.

\(^\text{10}\) In reality, when encoding real video the compression factor is not constant but variable since the amount of data produced by the encoder is a function of motion. The figures do give a good estimation of what can be achieved however.
Table 2.4

<table>
<thead>
<tr>
<th>TIME</th>
<th>COMPRESSION RATE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>None</td>
</tr>
<tr>
<td>1 second</td>
<td>27.65MB</td>
</tr>
<tr>
<td>1 minute</td>
<td>1.65GB</td>
</tr>
<tr>
<td>1 hour</td>
<td>99.53GB</td>
</tr>
</tbody>
</table>

Significantly with digital video, it is possible to dramatically reduce the amount of data generated even further by reducing the perceived frame rate of the video from 30fps down to 15fps or lower. This can be achieved by explicitly limiting the number of frames or through a bandwidth limitation mechanism. For multicast conferences on the MBone, for example, the bandwidth used is between 15 and 64Kbps. Although reduced frame rate video loses the quality of full-motion video, it is perfectly adequate for many situations, particularly in multimedia conferencing.

2.4.3 Slow Server Response Times

From a user perspective, it is fruitless having a high-speed connection to the internet if the server one is attempting to access multimedia information from has a slow response time. Unfortunately, many servers hosting multimedia applications are not planned properly initially and so run into problems later on.
One issue often overlooked is to ensure that key server processes such as the name server process (DNS) and the mail agent (SMTP) get enough cycle time to permit other users to continue operating. This can be done by tuning the operating system or some Web managers suggest it is sometimes best to distribute these high priority processes to stand-alone servers and dedicate another server to the multimedia application.

The second goal is to get enough disk throughput to accommodate parallel, high bandwidth requests. Two mechanisms which enable high disk performance to occur are as follows:

- Rapid Array of Inexpensive Disks (RAID) - essentially whenever a RAID disk controller receives instructions from the host to write a block of data to the array, it breaks that block of data into smaller pieces between all the hardware drives in the array. When the host requires data from the array, the controller simultaneously fetches the data from many drives. Parity is stored on another drive and this feature enables the user to reconstruct the data in the event of a mechanical failure in one of the participating drives. Six levels of RAID exist. RAID 3, for example, calculates parity information on every bit of data written to the data disks and writes this information to a dedicated parity disk, allowing for complete reconstruction of the data. RAID 5 is the most sophisticated implementation and allows optimised disk reads and writes for the greatest overall throughput.
• Hierarchical Storage Management (HSM) - this is more a concept then an actual product. The idea is to move the most frequently accessed data onto the highest performance storage medium. On the other hand, less frequently used data is migrated to slower storage devices. The criteria for actually selecting data to be moved and the storage path which the data follows as it becomes more or less popular are defined by the system manager. HSM systems can automatically transfer certain files from hard drives to secondary storage devices such as magneto-optical jukeboxes or tape autoloaders. Thus, files which might not have been accessed for a few months, say, can be automatically transferred to secondary storage. A marker is left on the primary storage device for the file so that if a request arrives to open a file which has been replaced by a marker, the HSM automatically and transparently migrates the file back to the server.

Some multimedia servers also make use of recordable CD-ROMs (CD-R) which can contain multimedia information in a compact format. These CD-ROM enable the provision of multimedia data at transfer rates matching most of the currently available network bandwidth and are relatively inexpensive compared with RAID or HSM products.

2.4.4 Storage and Retrieval Efficiency

The whole area of multimedia storage and its subsequent retrieval is one of great importance since there is an obvious need to be able to compress very large multimedia
files into storable ones and to retrieve such compressed files with the minimum of delay and required quality. As a consequence, the storage of multimedia information will be the focus of this thesis and the various mechanisms by which multimedia data can be stored will be examined in some detail in the next chapter.

2.5 OMG CORBA Standard for Multimedia

Recently, the OMG has ratified a standard for the management and control of AudioVisual (AV) information streams. The standard, proposed jointly by IONA, Lucent Technologies, Siemens with help from Fore Systems Inc., marks the arrival of full interactive multimedia applications built around the CORBA standard. The AV streams standard will open the way for the fast creation of commercial interactive multimedia applications such as video on demand. At present the standard is not yet part of the official CORBA 2.2 standard specification but rather part of the OMGs ‘Work in Progress’. Its revision deadline is October 1998 and will become part of the standard if accepted [OMG 97].

2.5.1 Standard Overview

The “Control and Management of Audio/Visual Streams” submission provides definitions of the components which make up a stream and for interface definitions onto
stream control and management objects (interface 1a) and for interface definitions onto stream interface control objects (interface 2b) associated with individual stream endpoints (see Figure 2.7)

Figure 2.7

Essentially the document proposes a set of interfaces which implement a distributed media streaming framework. Principal components of the framework are:

- Virtual Multimedia devices and Multimedia device (VDev and MMDevice) interfaces.
- Streams (StreamCtrl interfaces)
• Stream endpoints (StreamEndPoint interfaces)
• Flows and flow endpoints (FlowConnection and FlowEndPoint)
• Flow Devices (Fdev)

2.5.2 Component Definitions

In order to fully appreciate the operation of the standard, its component parts need to be addressed and explained. This will also serve to illustrate how the individual components of the model link up.

2.5.2.1 Streams and Stream Endpoints

A stream represents continuous media transfer, usually between two or more virtual multimedia devices. A stream endpoint terminates a stream. A simple stream between a microphone device (audio source or producer) and speaker device (audio sink or consumer) is shown in figure 2.8. A stream may contain multiple flow endpoints. Both flow producer and flow consumer endpoints may be contained in the same stream endpoint. Flows may also travel in both directions within the same stream.
2.5.2.2 Multimedia Devices

A multimedia device (MMDevice) abstracts one or more items of multimedia hardware and acts as a factory for virtual multimedia devices (VDev). It is described by the OMG Interface Definition Language (IDL) interface MMDevice. A multimedia device may support more than a single stream simultaneously so, for example, a microphone device can stream audio to two speaker devices via separate, non-multicast connections. For each stream connection requested, the MMDevice creates a stream endpoint and a virtual multimedia device.
2.5.2.3 Virtual Multimedia Device Interface

A MMDevice can be connected or bound to one or more compatible multimedia devices using a stream. It can potentially support any number of streams to other multimedia devices. Each stream connection is supported by creation of a virtual multimedia device (VDev) and stream endpoint (StreamEndPoint) which represent the device specific and network specific aspects of a stream endpoint respectively. VDevs have configuration parameters associated with them, for example, a microphone device may be capable of encoding data via μ-law or A-law. The sampling frequency may also vary. When two VDevs are connected by a stream they must ensure they are both appropriately configured. It does this by calling configuration parameters such as

```
set_format("audio1","MIME:audio/basic")
```

and

```
set_dev_params("audio1",myAudioProperties)
```

on the Vdev interface for the speaker device. This occurs when the StreamCtrl interface establishes a stream between two virtual devices.

A MMDevice supports operations to set up a stream between two or more MMDevices on the network, for instance, to bind two MMDevices using a stream the application programmer could call the following operation on the MMDevice interface

```
StreamCtrl bind(in MMDevice peer_device, inout streamQoS the_qos, out boolean is_met, flowSpec the_spec);
```
where \textit{peer\_device} refers to the multimedia device which should be bound to the active
device via a stream and \textit{the\_spec} specifies the flows within the stream which should be
initially connected. \textit{the\_qos} refers to the quality of service of the multimedia device.
Essentially, this means different things at different levels. For instance, a multimedia
device which supports video will be concerned with \textit{application level} QoS parameters
such as frame-rate and colour depth. Similarly, \textit{network level} QoS for the underlying
network protocols may include such parameters as minimum bandwidth and jitter.
\textit{StreamCtrl} indicates the reference returned by the bind operation which the application
programmer may use to start, stop or otherwise manipulate the data stream.

\textbf{2.5.2.4 StreamCtrl Interface}

This interface abstracts a continuous media transfer between VDevs. It supports
operations to bind multimedia devices using a stream. Operations also exist to start and
stop a stream. If an application programmer requires more complex functionality (such as
a rewind facility) the basic stream interface may be extended to support this. The
application programmer may establish a stream between two devices by calling the
following operation on the StreamCtrl interface.

\begin{verbatim}
boolean bind_devs(in MMDevice a_party, in MMDevice b_party,inout streamQos
the_qos, flowSpec the_spec);
\end{verbatim}

In this case, QoS is generally an application-level QoS specification. Appropriate
protocols are chosen by StreamCtrl and translation occurs between application level QoS
and network level QoS for these protocols. If the MMDevices decide they can accept the connection, each will create a StreamEndPoint and VDev to support the stream between them.

### 2.5.2.5 Point to Multipoint Streams

The media streaming framework supports point to multipoint streams. These allow a single source device to be connected to many sink devices via multicast streams. For example, the following structure might be utilised to initiate a video broadcast

```c
MyStream->bind_devs(cameraDev,nilObject,someQoS,nilFlowSpec);
```

Here using the reference to a `nilObject` as the second multimedia device (the b_party parameter) has the effect of adding the camera device as a multicast a_party. The application programmer could then add an arbitrary number of television devices to the broadcast by calling

```c
MyStream->bind_devs(nilObject,TVDev,someQoS,nilFlowSpec);
```

### 2.5.2.6 StreamEndPoint Interface

This contains the flow endpoints for each of the individual flows in a stream. Two types of stream endpoint exist: an "a-party" and a "b-party" (Interfaces StreamEndPoint_A and StreamEndPoint_B respectively). Both parties can contain producer as well as consumer flow endpoints. The distinction between parties is made so that when an instance of
StreamEndPoint is created, the flows will be in the right direction i.e. an audio consumer FlowEndPoint in a StreamEndPoint_A corresponds to an audio producer FlowEndPoint in a StreamEndPoint_B. The device is arbitrary as to which end-point is the A or B party. A call to connect() establishes or binds the stream on either a StreamEndPoint_A or StreamEndPoint_B.

2.6 CORBA, WWW & Distributed Multimedia

Many experts [Mackay 96, Buford 96] in the field are of the opinion that the WWW has now effectively reached a critical mass, the major symptom of which is the bandwidth problem. Essentially, the Web has far surpassed the reach of any other distributed information architecture and it is believed that the time has arrived to commence the integration of CORBA and the WWW. The next sections discuss how this might be achieved and also examines work carried out in relation to distributed multimedia on the internet.

2.6.1 Building a Web Common Facility

As it stands, OMG CORBA’s forte comes from being an architecture for distributing structured information. In order to provide true interoperability between CORBA and the Web, interfaces used in Web application development would need to be standardised. For example, CGI in its present form does not scale to large or complex applications so HTTP server vendors such as Netscape and Microsoft have created their own (incompatible,
non-portable) APIs for HTTP servers called NSAPI and ISAPI respectively. With multiple vendors supplying different APIs for the same purpose, this would be ideal situation for OMG standardisation. It has been suggested, therefore that the OMG create a Web common facility which standardises these interfaces to enable portability of Web applications.

The aim of the Web common facility would be to provide well-defined interfaces which encapsulate the details of HTML and HTTP for Web application developers. This facility would enable the Web to link with high quality production applications seamlessly. Another benefit would be to facilitate the construction of CORBA HTTP servers which could take advantage of the CORBA common object services so vital for organisations using Web servers for information management. These would include naming, security, managing persistent objects, concurrency and transactions. One company called ANSAWeb [ANSAWeb 96] have experimented with similar ideas and have reported promising results - their project utilised IDL to encapsulate HTML.

A related topic is the mechanism which enables access to objects from HTML clients. A standard CORBA URL access method could be created, for example, of the form:

\[
\text{corba://objectName}
\]

A corba protocol might be used in such cases as distinct from the normal http protocol. This would bind the HTML client to the object objectName. A list of the possible operations which could be performed on such an object would then appear in the client in
an HTML form. When the user selects an operation (along with an argument list possibly) and issues a POST method, the Web common facility would marshal the arguments into a CORBA name-value list and invoke a standard operation on the object, passing the name-value list as the argument. The operation would then take an appropriate action and format the response back to the client.

The major advantage of this mechanism is that the HTML client itself requires few changes (if any) but still offers full access to CORBA objects. In addition no HTTP modifications are necessary in order to implement the facility and the fact that the objects are being accessed in the same way as all other Web resources, via URLs, means that HTML document authors are already familiar with the notation.

2.6.2 Constructing a More Object-Oriented Web

At present the object semantics supported by the Web are, at best, limited. Examples such as electronic commerce, digital catalogues and virtual 3-D environments illustrate clearly that current Web applications are becoming increasingly interactive. Current versions of HTTP, HTML and CGI do not easily meet the requirements for such applications and it is believed that these difficulties may be overcome by adapting object-oriented solutions to the Web framework. Support for such hybrid Web/distributed object applications necessitates alterations in the areas of client-side state management, embedded objects and scripting mechanisms as follows:
• Client-side state management - many Web applications require the management of state information for clients participating in transactions. HTTP is a stateless protocol thereby making this requirement difficult to satisfy. It is suggested therefore that future versions of HTTP be extended to incorporate the management of client-side state.

• Embedded Objects - being able to embed objects into HTML documents with the view to providing dynamic content has already got several existing approaches in place. Netscape Navigator, for example, supports Java applets to be run within the browser via the <APPLET> tag and Microsoft’s Internet Explorer provides built-in support for VRML. The problem here is that no standard mechanism for embedding such objects has been clearly defined and thus many experts believe that the approach specified in the Compound Document ought to be adopted to standardise the embedding mechanism.

• Scripting Mechanisms - Currently Netscape, with JavaScript, and Microsoft, with Visual Basic Script (VBScript), have added script language capabilities to their browsers. This facilitates a finer level of control for HTML clients and reduces the need for external communication by embedded objects thereby decreasing server interaction. A standard HTML scripting mechanism, it is suggested, should be put in place which would identify the capabilities and constraints of scripting languages without actually specifying a syntax for the languages themselves.
2.6.3 Distributed Multimedia Systems

Some reports have suggested that successful multimedia application development has suffered due to the ambiguity existing in the term 'multimedia' [Wallis 95] while others blame the fact that programming languages, operating systems and compilers do not "understand" multimedia data very well [Smith 96]. Despite such reports, however, much research is currently being carried out in the area of distributed multimedia systems and it is seen as the next generation technology for computers and communications networks [DMSL 96]. One of the forerunners in such research has been the Distributed Multimedia Systems Lab (DMSL) at the University of Massachusetts which concentrate research efforts in the area of multimedia information systems.

2.6.3.1 Multimedia Information Systems

As distinct from traditional text-based systems, multimedia information systems (MMIS) offer the possibility of bringing the non-text media on line. A good example of what has been achieved in this area is the Hypermedia/Time-Based Structuring Language (HyTime), now an ISO standard, which essentially specifies how certain concepts common to all hypermedia documents may be represented using an extension of the Standard Generalised Markup Language (SGML) which deals only with textual documents [Koegel 93]. These concepts include the association of document objects with hyperlinks and the placement and interrelation of document objects according to coordinate systems which can represent space, time or any quantifiable dimension. HyTime
also provides a means for defining the representation of hyperdocuments as character files which can be interchanged between and processed on any platform.

The developers of HyTime have also created an architecture for a MMIS which stores such HyTime documents. The system includes a backend SGML parser, an object-oriented database to store document description and a HyTime engine called HyOctane for access to and presentation of document instances. A HyTime-conforming document type definition (HDTD) for encoding interactive multimedia presentations has also been created. The architecture for the overall system is shown in Figure 2.9.

Further work in the area has seen the development of a model which integrates object-oriented scripting languages for multimedia applications such as OpenScript (in Asymetrix' Toolbook) and Lingo (in MacroMedia Director) with HyTime [Buford 94]. Such an infrastructure allows multimedia document authors to add interactive behaviour to the application without having to resort to conventional software development techniques.

Also, approaches have been proposed which allow the basic distributed hypermedia document format, HTML, to be processed within SGML and HyTime systems. This prevents the obsolescence of HTML documents which currently fail to meet the needs of open and integrated hypermedia environments, vital requirements for distributed hypermedia documents [Rutledge 95].
In addition work has been carried out which allows HyTime documents to be presented by converting them to HTML for processing by HTML browsers [Rutledge 96]. Although many institutions are using it to encode their documents for WWW hypertext presentation, HTML enforces a single document that is closely tied to its presentation. This is not a satisfactory format for the long-term storage of hypertext documents that may be presented by means other than HTML browsers. HyTime encodes the presentation-independent structure of hypermedia documents and is therefore more appropriate for long-term storage of hypertext documents.
Its creators believe it will play a significant role in the development of future MMIS due to its close relationship with SGML, its potential use as a data model for multimedia information and its role as a multimedia interchange format.
2.7 Web Server Application Design using Advanced Software Technologies

The next section details the development of a Web Server application which utilises some of the latest internet and WWW technologies.

2.7.1 Web Application Architecture

This application comprises a number of distinct parts, some of which need to be discussed in some detail in order to fully appreciate both their inclusion and their role in the system under discussion. Essentially the overall system is made up of the following subsystems:

- Web server software (Apache HTTPd) residing on a remote UNIX sparcstation (skelig)
- Web client software (Netscape) residing on a local UNIX machine (einstein)
- a client HTML program (Noticeboard.html)
- a CGI program (notice.cgi)
- a Java application (NoticeboardApp.java) which uses a JDBC client (I-Kinetics OPENjdbc driver) and Databroker 5.0 JDBC Server [I-Kinetics 97] to access a relational Oracle7.0 database table (staffaccounts) (see figure 2.10).

The complete coding for all parts of the application is located in Appendix A at the end of this thesis.
2.7.1.1 JDBC

JDBC, often incorrectly referred to as Java DataBase Connectivity, is a Java API for executing SQL statements [JavaSoft 97]. It consists of a set of classes and interfaces written in the Java programming language. It is a tool for database developers and makes it possible to write database applications using a pure Java API. Using this mechanism it is possible to send SQL statements to almost any relational database. In this way it is
feasible to write a Java application which uses JDBC once and it can be used on a multitude of different databases on many different platforms. Hence, an enterprise could use JDBC to connect all its employees, possibly on various platform types, to one or more internal databases via an intranet.

JDBC achieves three fundamental goals:

- establish a connection with a database
- send SQL statements to the database
- process the results received

Much work has gone into the creation of applications which make accessing a database more user-friendly and easier for the end-user e.g. an application might present a menu of database tasks from which to choose. After a task is selected, the application presents prompts and blanks for filling in the information needed to carry out the selected task. With the requested information, the application then automatically invokes the necessary SQL commands. With such an application, users can perform database tasks without any knowledge of SQL syntax. This sort of approach is inherent to the system being considered here.

One important point that needs to be mentioned is the fact that Microsoft's ODBC (Open DataBase Connectivity) API is the most widely used programming interface for accessing relational databases on all platforms. Some of the reasons why JDBC is superior to ODBC include:
• ODBC is not appropriate for direct use from Java since it uses a C interface. Calls from Java to native C code have a number of drawbacks in the security, implementation, robustness and automatic portability of the application.

• ODBC is hard to learn and has complex options even for simple queries whereas JDBC keeps simple things more straightforward.

• A Java API such as JDBC is needed to enable a pure Java solution. When ODBC is used the driver managers and drivers must be manually installed on every client machine which is cumbersome. When the JDBC driver is written completely in Java, the JDBC code is automatically installable, portable and secure on all Java platforms.

2.7.1.2 JDBC Drivers

The JDBC portion of the Java application makes use of a DriverManager object in order to load the required JDBC driver for connection with the database. The type of JDBC driver used to connect to a database is an important consideration for a Web application and can fall into 3 broad categories as follows [Cottman 97]:

• **Client-Centric** which is located entirely on the client system, converts JDBC calls into calls on the client API for relational databases. This approach takes advantage of the wide availability of ODBC drivers but is limited in the sense that binaries are needed on the
client and for security reasons many browsers do not allow access to foreign binaries. In addition client-centric JDBC drivers are not pure Java implementations and have a very high cost of ownership, requiring installation, administration and maintenance for each client and may require frequent installations to remain up to date.

- **Network Centric** which are divided into two separate parts: a 100% Java JDBC client and a JDBC server (which may or may not be 100% Java). The client can pass SQL commands over the network to the JDBC server and get information back from the server. This comprises a three-tier architecture as such consisting of JDBC client, JDBC server and DBMS. However, whereas most of the client's code is concerned with transforming network data received from the server into Java data types, the JDBC server handles direct connections to the database as well as handling any exceptions which may arise from SQL execution. It also 'packages' this data for transmission over the network to the JDBC client. This architecture has the typical three-tier advantages such as access control, load balancing and encryption and also has a lower cost of ownership than a client-centric JDBC driver as the DBMS drivers only need to be located on the JDBC server platform meaning less maintenance and administration is necessary. The major drawback of this approach is firewall security in the sense that each JDBC implementation requires the creation of a new proxy since the protocol exchange between JDBC client, JDBC server and DBMS is unique. If a proxy is not available for the particular brand of JDBC being used then firewall security must be relaxed if the JDBC application is to be able to access a database behind the firewall. In addition, since this
approach typically requires proprietary middleware, it loses out on all the benefits associated with open technologies such as CORBA and IIOP.

- **CORBA-Based** which is an integrated approach, building on one of the drawbacks of network-centric JDBC drivers - the fact that much processing is spent in handling the communications between its distributed client and server components. The CORBA-based mechanism uses the CORBA and IIOP infrastructure where processing can take place across a variety of systems, thereby improving the quality of service of the JDBC driver.

CORBA, discussed in the last chapter, provides a way in which objects written in almost any language located anywhere on the network can communicate with each other thereby allowing users to access a wide variety of computing services. It also defines a language, IDL, used to define specifications for CORBA-compliant objects. CORBA objects themselves communicate through Object Request Brokers (ORBs) which reside on all CORBA clients and servers and these ORBs communicate with each other over the network using the CORBA Internet Inter-ORB Protocol (IIOP) to give users access to CORBA services.

Hence, through basing a network-centric JDBC driver on CORBA, the driver can become an assembly of collaborating objects rather than a single, large application residing on a server. Through IDL it is possible to create a CORBA-compliant specification to the JDBC server objects and these can be implemented using Java (or
other language for which an IDL compiler exists). The server objects can then be made available over the network through IIOP and reused by other applications. In addition, a JDBC server implemented as a CORBA object can package multiple databases into a single server with a single IDL interface giving a system and language independent interface to the JDBC server usable by all other CORBA-compliant servers and clients.

In the application under discussion here, for the reasons cited above, the CORBA-based JDBC driver approach has been chosen and the implementation used, developed by I-Kinetics is called OPENjdbc. This is a 100% Java client which connects through IIOP to the I-Kinetics DataBroker5.0 server (this comes with Iona Technologies' Orbix runtime classes), which provides database access services. DataBroker itself is CORBA-compliant and multithreaded so it can handle multiple database connections simultaneously. Since the DataBroker server has an IDL specification, this means that the services it provides can be made available to all CORBA-compliant clients and servers on the network. It has the additional advantage of supporting ODBC drivers and may be used to access a variety of databases without depending on database specific protocols. Similar to other driver types, the OPENjdbc driver can be used in a Java application through the DriverManager object - allowing us to create a database application which is truly 'open'.

The only major drawback with this particular approach is that IIOP does not work with firewalls so security is still a problem. HTTP tunnelling must be used to enable IIOP transactions to occur across a firewall but this can reduce network performance. That
said, however, many firewall vendors are currently working to support IIOP as another standard protocol so this should not be a problem in time.

2.7.2 System Operation-The Client

What the end-user sees in this application essentially consists of an HTML file called Noticeboard.html which has been implemented using HTML 2.0. This program runs in the Netscape browser. The Web page itself acts as a potential gateway to a noticeboard-type utility which can be used by staff at ITT. Any staff member who has been set up with an account for the Noticeboard System can simply supply the appropriate username and password combination, click the login button on the Web page and become connected to the Noticeboard utility where they could potentially read or submit their own notices for general viewing. When a true account holder logs in, the system responds with a welcome message but if a username/password combination is submitted which fails to correspond with those stored in the legacy Oracle database on the server machine then the user is given an appropriate warning message on their browser as a response.

From a purely multimedia aspect, the web page consists of a number of still images in both GIF and JPEG format. Some are accessed from the local machine while others are accessed via the network from other ITT servers (namely the machines called brandon and ppc). The ITT logo, for example appears on the Web page in JPEG format. Also included on the page are two animated GIFs which give the page “life” and make it more interesting.
2.7.3 System Operation - The Server

When the user types in their username/password they are effectively filling out an HTML form. When the login button on the Web page is clicked the Apache HTTPd server software automatically activates a CGI program (notice.cgi) located in the /cgi-bin directory on the server machine. (The exact URL to this location is known in advance and set up in the HTML file). The CGI script itself acts as an interface between the HTML form and the Java application (NoticeboardApp.java) - its functions are to commence execution of the Java program and pass it a list of the necessary server environment variables so that it can do its job.

The Java application is at the centre of the system and provides several important tasks as follows:

- It adds the I-kinetics OPENjdbc client driver to its jdbc.drivers property list
- It connects to the Oracle database using the OPENjdbc driver and CORBA Databroker server combination via CORBA IIOP. An URL which will allow connection to Oracle, along with an Oracle account/password is supplied for connection purposes.
- It sends an SQL query to the table staffaccounts in the Oracle database and gathers all the results in a ResultSet object.
- It parses all the HTML form information into a hashtable.
• It compares the username/password information in the hashtable with the information in the ResultSet object.
• It catches any errors or exceptions from all possible sources.
• It generates appropriate HTML-type formatting for the response
"C makes it easy to shoot yourself in the foot. C++ makes it harder, but when you do, it blows away your whole leg."
Bjarne Stroustrup, creator of the C++ programming language

As the internet has grown in popularity, so too has the diversity in the types of data used both on it and, more especially, the Web. The previous chapter presented a number of the most important Web multimedia data types and how they are dealt with for visual or audio purposes once they have been retrieved from the hosts on which they are stored. This chapter discusses the most popular mechanisms by which the information used on the Web is stored. It details specific examples of each approach and lists their relative pros and cons. In particular it focuses on multimedia storage options and draws conclusions concerning the most appropriate techniques to utilise for the storage of such data.
3.1 Introduction

Historically, the storage of information has always been important. Astronomical and meteorological records, for instance, were documented on parchment in by-gone eras. Nowadays, with the widespread availability of computers and electronic storage mechanisms, a complete shift has taken place and the storage of information on paper has decreased significantly. In modern businesses, such as banks and manufacturing industries, most pertinent information is stored electronically in some form.

The arrival of the WWW has created an information explosion and many organisations are attempting to take advantage of the global customer base which the Web provides. This involves the creation of a Web site where the company can advertise their products or services and possibly provide capabilities such as order forms where a customer can order goods or services via the Web and even pay for such items. The most important consideration for such Web sites is information storage. Every commercial Web application will have its own characteristics and so the required storage techniques will vary from case to case. However, there is currently a shift to the incorporation of less traditional information, especially multimedia data, in such applications and this can have major repercussions on the type of storage technique required.
3.2 File System Storage

3.2.1 Background

The first applications used to manage large quantities of data were developed in the 1960s and these held their data using file system storage. The mechanism is simple and especially effective for applications where no collection of records need to be kept in a back-end database (in the 1960s the database concept had not yet arrived so system developers who needed to store record collections had no choice but to use files). Such applications are appropriately served by the data model available in the file system, that is, a sequence of characters of arbitrary length. These systems made use of the standard file access methods via programming languages such as Fortran, PASCAL, C and COBOL. A traditional example of a system using this approach might have details of an ordering system stored in plain text files on the hard disk of the machine utilising a file organisation such as sequential [Severance 72], indexed sequential, indexed [Lum 73] or direct [Dean 71]. These basic file designs are still in use and each type has its own unique suitability to certain applications in terms of appropriateness and performance. In modern times, the file system approach is used more increasingly for storing the multimedia content associated with Web sites. Other common applications which may use this storage approach successfully are the common text editors such as Notepad on Windows systems or vi on as they are essentially ‘non-query’ type applications with no requirement for query languages.
3.2.2 Pros and Cons of the Approach

The advantages of this approach include

- simplicity - there is no database of any description involved and hence, no query language needs to be learned.

- high performance - a file system invariably has higher performance than any more sophisticated system.

- maintainability - for small applications, providing there are no modelling alterations required, there is the minimum of effort involved in keeping the software in perfect working order.

The disadvantages include

- directory structure alterations - in Web based applications, a change in the directory structure can mean the URL of some or all multimedia data being referenced needs to be updated manually which is both time consuming and tedious.

- removal of content - in Web applications, resources being accessed remotely can be placed in a different location on the host machine or worse still, removed completely.
from the host without the knowledge of the local HTML author. This can lead to a totally ineffective web page.

- data inaccessibility - in traditional file system based applications created using high-level programming languages the only way to access the information stored in the text files is through the associated file management operations which are limited and completely procedural i.e. there is no high-level data manipulation language (DML) to query or inspect data easily.

- data dependance - for traditional applications which model and store data structures of some kind via a high-level programming language such as C there is considerable effort involved in updating the system if a change is made to the stored data structure. Also these systems need to know how the files are structured and the methods for accessing them. If the way in which the files are structured needs to be modified, then the application also needs modification. For example, if an original data structure has a print() method which outputs the details of an entity and this entity is modified in some way then the print() function will also need to be altered. In addition any files which process this entity's data structure, for instance a function which gets details about the entity from the user, will also need modification. Hence it can be seen how inflexible the file system storage mechanism is for such applications.
Chapter 3 - Storing Multimedia Objects for use on the Web

• Lack of shareability - it is difficult to share data between several applications or users since each user needs to see the latest up-to-date information as though they alone were accessing it. This is not feasible with the approach and access conflicts will occur.

• Case-sensitivity issues - file names can be a problem if, for example, the externally accessed objects reside on a Windows based machine where case-sensitivity is not an issue. For example, FILENAME.gif and filename.gif and FiLeNaMe.gif refer to the same file on a Windows machine. If the actual name of the file is filename.gif and the author is on a UNIX system (which is case-sensitive) and uses FILENAME.gif to reference the graphic object then it will not be accessible.

3.2.3 Sample File System Storage Application

A good example of where a file system is used for storage arises for a simple Web personal 'homepage'. In this case the author gives textual information outlining some personal details. In addition some multimedia objects are normally incorporated to give the homepage some aesthetic appeal. In this instance the most effective approach to take is to store the multimedia objects in the file system itself, perhaps in a directory named after the type of object involved such as a Gif directory for GIF images, Jpeg for JPEGs and so on. Figure 3.1 shows a homepage which incorporates a number of links to other Web locations. In addition a GIF image is incorporated into the page to give it "life"
Hi, my name is John Brosnan and I am a Masters student at the Institute of technology in Tralee. The title of my research is "The Application of Advanced Software Techniques to Internet and WWW Server Development". In addition, I am a temporary-wholetime lecturer in the Department of Computing at the Institute.

A major part of my research involves the investigation of Object Database Technology and in particular, Computer Associates Jasmine ODBMS product. Also, I have studied the OMG CORBA standard for distributed systems and used the Iona Technologies OrbixWeb CORBA product in my research.

For relaxation I enjoy reading, films and watching the Simpson’s. I also enjoy listening to music, especially 2FM.

A Web Homepage Storing its Multimedia Content using File System Storage
3.3 Database Storage Mechanisms

3.3.1 Background

In the early years of data processing, considerable effort was put into improving the physical file supports and developing techniques for accessing files and records. However, the data that needed to be held in order to manage an organisation was becoming increasingly complex and interdependent. Users of various data files wanted to integrate files and applications which had been previously used independently. They also needed to be able to represent more complex links between the various records. Traditional file management systems were inadequate in this respect and, hence, the concept of the file was gradually replaced by that of the database.

The term 'database' can be defined many ways but when talking about the normal case of an organisation or global audience an appropriate definition is as follows:

'A database is a large integrated, shared pool of information in a form suitable for handling by a computer which is a basis upon which the computer user community within an organisation can draw inferences in conducting its business.'

3.3.2 Database Management Systems - Function and Goals

A Database Management System (DBMS) is the software package used to interact with the database. It can be used to define data, consult the database or update it. An essential aspect of such systems is that they must allow users to specify the data they wish to find
in abstract terms, while at the same time undertaking the search according to how the data is organised in the database. Nowadays, many types of DBMS exist (these will be dealt with in the remainder of this chapter) but the fundamental requirements of all include [Delobel 95]

• Shareability - the need to share data between several applications is one of the main reasons for the formulation of the database concept. Different applications working with the same data must be able to execute as if they were the only one working with that data. The DBMS must provide the means for managing data sharing and for detecting any access conflicts that may arise between several users or applications and provide the tools for resolving them. From a Web server perspective, there must also be a facility whereby two users, perhaps thousands of miles apart, may access the global Internet simultaneously in order to process or 'share' the same database in some way.

• Consistency - In a multi-user environment where information is shared simultaneously, such as databases which reside on Web servers linked-up to the internet, it is vital that the database presents a consistent picture of the operations of the enterprise it is modelling to all users at any given moment in time. DBMSs ensure consistency via a locking mechanism which prevents two concurrent users from accessing information in conflicting ways.

• Reliability - although computer hardware has become increasingly more reliable over the years, it does break down occasionally and the database must be protected against
such incidents. At the very least this means taking regular dumps of the database from its normal storage on magnetic disk to a back up magnetic medium - normally magnetic tape due to its relative cheapness. This process is of paramount importance, especially for commercial databases, such as the international stock exchanges, which exist on the Web and elsewhere.

• Ease of Access to Data - A DBMS must allow any item in the database to be accessed easily. More specifically, the system must allow data to be accessed using high-level declarative (non-procedural) languages called *query languages*. Such languages contrast sharply with the classic file management operations which are limited and totally procedural. Users may use query languages interactively to consult or modify a database. There are therefore two different ways of accessing a database: in an application connected to a DBMS (i.e. a user program where query language statements have been embedded) or interactively, using the query language. On the Web the user is often in a position to create these queries *dynamically* just by entering details into list boxes and hitting a "query" button on the Web page. The Web application developed in chapter 2 achieves this functionality.

• Security and Privacy - security means security of the database contents from accidental or malicious damage. This implies that there must be control over access to the database. Privacy means that the database contents should be available only to those who need it in order to perform their jobs properly and who are given the appropriate level of database access. The permission issuing authority is normally the Database Administrator (DBA)
who decides what level of access an account holder requires. In terms of databases which reside on Web servers similar rules apply. In this case the Web server administrator is usually the DBA also and can decide what persons may access the database, whether or not it can be accessed only internally in the organisation via LAN or globally on the internet and what type of access level each user will have.

• Data Independence - this a major advantage provided by a DBMS. An application which handles data using a file system is strongly dependent on its data. The application must know how the files are structured and the methods for accessing them. If the way in which the files are structured needs to be changed, this cannot be done without requiring modifications to the application. In contrast a DBMS should allow applications to be written without the programmer having to worry about the physical structure of the data and the associated access methods. Hence, the system can evolve to take account of new needs without disturbing applications that have already been written. A good example of this is the case on the WWW where an end-user attempts to access a database which resides on a Web server. Even though the database structure may change internally in some way, the Web page or application which "hooks up" to the database for should not need be altered in any way.

• Availability - an organisations database must be available to the user community when and where they require access to it. As these users could be scattered over one large geographical site or over many such sites at many different locations, it is inevitable that some form of telecommunications network must be used to support database access. The
simplest arrangement, for users all located at one private geographical site is the Local Area Network (LAN). Here one computer system (or there can be others) serves as a database server within the network. Users may be directly connected to the network or connected to other computers which are in turn connected to the network. Regardless of how they are connected, each user has transparency of access to the database within the limitations only of the access control permissions allocated to them. Nowadays, the LAN network will have a bus or ring topology and will conform to some standard, such as the Ethernet LAN standard, in its implementation.

For users scattered over many geographically separate sites, or if the single site spans a large area, then some users will have to be connected to the database server via the public telecommunications network, generally known as a wide area network (WAN). The internet is, by its very definition, a worldwide international network spanning the entire globe and so comes under this banner. The connection from LANs to WANs is effected by the use of devices called communications gateways. Again, the location of the database server is completely transparent to the end user so that a person in London accessing a Web database server which resides in New York will see the same "picture" of the database as a person in China provided they have equal access to the database. Full support for LAN/WAN networking is typically provided by a DBMS product. The computers used in the network itself may be homogeneous (all of the same type) or heterogeneous (from diverse manufacturers). Again, DBMSs normally support homogeneous and heterogeneous working across networks.
• Flexibility - As a general rule, most organisations cannot anticipate what all its information needs for the future will be. Thus, it must be possible not only for a database to grow in terms of the volume of information it contains at any time but it must also be capable of accommodating new types of information as time goes on. This is especially important in today's marketplace where most organisations are aiming to establish a presence on the WWW and the use of multimedia information is becoming essential in the creation of a "good looking" Web site. Equally, organisations will not be able to foresee the uses they will put their database to i.e. the software applications which will be developed to make use of it. Hence, it must be possible for a variety of programmed applications written against a database to grow over time.

• Database Administration - The database administrator plays a major role in the design and maintainence of a DBMS. Its main functions include deciding what data is to be stored in the database, deciding on the physical structures and the access strategies, defining permissions granted to users, performing backups of data and optimising the physical organisation to improve the system's overall performance. The DBA for databases linked to a Web server must also consider global access and the extra security implications it may have for the database itself and also the organisation's internal network or intranet. There is also the fact that the database is open to a worldwide audience of billions of people and so the network must be able to handle the anticipated level of queries to the database so there must be interaction between the DBA and network administrator (who is often the same person anyhow).
3.3.3 Representation Levels in a DBMS

The concept of data independence is a fundamental aspect of a DBMS and in order to attain such independence, three levels of representation are considered: the physical level, the conceptual level and the external level [DTG 75] (See Figure 3.2).

Figure 3.2

Levels of Representation within a Database
3.3.3.1 The Physical Level

This level defines the databases physical schema, that is, how the data is represented on the storage device used by the computer system. The physical schema is defined in terms of files and records. The computer’s file management system and operating system are responsible for the actual management of peripherals and devices.

3.3.3.2 The Conceptual Level

The purpose of the conceptual level is to define and manage the conceptual schema, which is the logical representation, inside the system, of the reality which the database is supposed to transcribe. The conceptual level is therefore the main part of the DBMS. Defining the conceptual schema is a modelling process, since it involves translating real-world entities into abstract terms. The DBMS provides a data model for this process, with an associated data definition language (DDL) which is used to specify the conceptual schema within that model. The data model has very important repercussions on the type of application a DBMS can support and on how those applications are built. In fact the model defines the system’s vision of the real-world. Applications have to use the possibilities the model provides and put up with its constraints. Three main models have been used in existing DBMSs. These are
• The Hierarchical Model - here data is organised in a hierarchical tree structure (See figure 3.3). DBMSs based on the model are still widely used today as they are ideal for applications such as online transaction processing (OLTP).\(^\text{11}\)

Figure 3.3

![Hierarchical Tree Structure]

- The Network Model - The network database model may be equated with a network file structure as shown in the Figure 3.4 [DTG 71]. These are used primarily for OLTP applications for which they are best suited.

\(^{11}\) Such applications include airline reservation, inventory systems and data warehousing
• The Relational Model - although the relational model was not the first to be proposed as a definition of a databases conceptual schema, it has become the one which has been used predominantly in modern commercial systems and today the relational model and the systems that use it dominate the database market. This is in no small part due to the simplicity of the model compared with the hierarchical/network models which preceded it. It achieves a level of abstraction from the underlying complexities of computer storage which are reflected in the other models and is one of their main problems. The model is based upon the mathematical idea of a relation and allows data to be represented in the form of tables whose size is predefined[Codd 70]. Figure 3.5 shows an example of a simple relation built on three domains: town names, population and country names. Here a town is modelled as a row in a database table. Each row is called a tuple and each
element within a tuple represents one of the town’s characteristics (name, country or population). To make representation of relations more meaningful, columns are named and these names are termed *attributes*.

**Figure 3.5**

<table>
<thead>
<tr>
<th>name</th>
<th>country</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paris</td>
<td>France</td>
<td>2600000</td>
</tr>
<tr>
<td>Madrid</td>
<td>Spain</td>
<td>1200000</td>
</tr>
<tr>
<td>Brussels</td>
<td>Belgium</td>
<td>1300000</td>
</tr>
<tr>
<td>Berlin</td>
<td>Germany</td>
<td>1000000</td>
</tr>
<tr>
<td>Marseille</td>
<td>France</td>
<td>1000000</td>
</tr>
</tbody>
</table>

*A table based on the relational model*

In the *TOWNS_TABLE* above the attributes are name (of town), country and population. An important point is the fact that the actual *position* in the table of rows and columns is of no importance. If the position of rows or columns were changed in the table above there would be no change to the information content or accessibility of the data. Hence, access to data in a database constructed according to the relational model has no dependence on positional information whatsoever unlike previous models. Also important is the fact that the tables must be *flat*, that is, duplicate columns are not allowed in the model.

The relational model includes a definition of certain fundamental operations which can be carried out with a relational database. These are incorporated into the international
standard for a relational database language providing both DDL and data manipulation language (DML) capabilities, the Structured Query Language (SQL).

3.3.3.3 The External Level

Normally, a DBMS is used by several clients simultaneously. The conceptual schema represents all the information known by the system, but individually each user normally only employs a small percentage of that data. Hence, the DBMS provides, at external level, the concept of the view (or sub-schema) which allows users to be shown the part of the conceptual schema that corresponds to their needs (or access rights). A view can also provide a user with data synthesised from the data actually represented in the database. For example, a view could give statistics about a set of people about whom the conceptual schema had detailed data. In the relational model, the view is a table.

3.4 Relational Databases

3.4.1 Background

Currently, relational databases are the most widely used type of database storage mechanism and, it appears, will continue to be for some time. As the name suggests, this type of storage mechanism is based on the relational model discussed earlier and has been so successful due to its mathematical basis and the simple way in which it represents information by the well understood convention of tables of values. Commercially available DBMS products based on the relational model appeared in the late 1970s and sales of these only took off in the early 1980s when the impact of improved hardware
chip technology caused a dramatic improvement in computer price to performance ratio. RDBMSs were also increasing in popularity since they introduced the so-called ‘Fourth Generation Language’ (4GL) which helped to reduce the costs of developing the many types of software application which made use of relational databases.

3.4.2 Pros and Cons of the Approach

The advantages of the relational database approach include

• Simplicity of the concepts and the schema - In the relational model the conceptual schema is made up of a list of tables which is highly intuitive.

• A good theoretical basis - Until the relational system was invented there was no formal theory for databases. This prevented any significant advances in research in the area. The relational model fulfilled this role by proposing a simple theory which allowed database research to develop in the same way that programming language research had already done.

• High-level manipulation languages - relational languages like SQL define the data you want to extract from the database, and not the way it is retrieved. Being a high-level non-procedural language means it is also easy to use.
Chapter 3 - Storing Multimedia Objects for use on the Web

- An improvement in integrity and security - Systems using the network or hierarchical models have inherent problems of data security. Relational systems, by using high-level languages and specifying integrity constraints, help in this respect.

- Manipulating data in sets - in contrast to first-generation systems, the relational model provides DMLs that allow sets to be manipulated globally. This allows duplicates to be managed automatically and makes it possible to use parallel processing to handle large sets.

- Standardisation - relational databases support the standard SQL-92 data model and so applications are portable.

- Flexibility and Extensibility - the relational model does not use positional information so it is possible to modify and extend the database to cater for new user requirements without necessitating change to programs already accessing it.

The problems associated with the approach are as follows

- an oversimplified model - traditional applications based on the relational model used only data with simple structures, built from fundamental types such as integers, reals and strings. In this respect the relational model proved successful. New applications, such as CAD and office integration, call for data structures with a more expressive power. Such
applications require hierarchical data structures which are not easily represented by the relational model.

- Another problem in this context involves data manipulation. Often a set of relational join operations are required and this can be very costly in terms of processing time, especially in the case of CAD where accessing an object can require several dozen joins.

- Limited manipulation languages - Relational DMLs do not allow an application to be programmed in its entirety and must be embedded into general programming languages. The encapsulation of one language in another can give rise to system malfunctions. At the programmer level two languages with very different paradigms must be handled, SQL being declarative and, for example, Java being procedural. Hence, debugging such applications can be long and difficult and the difference between the relational data model and the programming language type system gives rise to the so-called impedance mismatch problem which is one of the major bottlenecks for database application development using traditional database systems.

- Lack of support for querying multimedia data - recently many RDBMSs have added support for Binary Large Objects (BLOBS) which allow the storage and retrieval of multimedia data types. However, there are no provisions for querying the content of the BLOB. In addition the model can support only a limited amount of such information without degrading performance.
3.4.3 Sample Relational Database Application

The analysis and design of a relational database is essential if one is to implement a system which will perform to required expectations. In addition it is necessary to understand concepts such as tables, keys, relationships, integrity rules, information models and normal forms\(^\text{12}\) along with the relational model operators including join, division, projection, union and Carthesian product. To discuss the theory of relational database design would take a volume in itself [Occardi 92, Wiederhold 83] but the basic concepts involved can be seen from the following sample application.

3.4.3.1 Modelling the Information

The Institute of Technology in Tralee required an on-line system by which both students and lecturers could be informed about positions which were arising in industry and elsewhere. This would be of huge advantage to the institute since it could monitor the types of jobs which were appearing and keep contacts, through the Graduate Placement Office, with those companies using the system to advertise their positions. Each new job entered into the database would be matched to an employment category, which would be then matched to a course running in the college. This enables useful statistics to be generated to aid in future course development. The fact that the system would be online would mean that the student could use the system from any machine connected to the

\(^{12}\) Tables in a relational database must conform to 'normal forms' in order to eliminate certain update, insertion and deletion anomalies from the design of a relational database.
internet 24 hours a day, putting their CV details online, accessing employer profiles, employment vacancies and also making use of a noticeboard facility which would keep students informed of further education opportunities and other career-related Web sites. All students in graduating classes are shown how to use the system in order to avail of the opportunities presented.

The central component of TOPS is a search engine which allows the CV database to be queried on a set of selection criteria specified by a prospective employer. Candidates can be selected on criteria such as qualifications, key skills (languages, computer skills etc.), work experience etc. This is the single most important benefit of TOPS - when an employer contacts the College with an employment opportunity, a set of candidate CV's, all matching the employer's selection criteria, can be forwarded within a short space of time to the employer. This fast response time will give the ITT graduates a valuable edge in a competitive job market.

The screen capture in Figure 3.6 illustrates the home page of the system.
From the point of view of modelling the information used by the system, the Common Gateway Interface (CGI) was utilised to interface with the HTTP server used to store the system's data. The CGI programs in this system were written in C and contained embedded SQL calls in order to access the Oracle relational database which stored the system information. At the client side, HTML "fill-out" forms were used in order for interactivity to occur between the student and the system. Upon supplying the requisite
information the form returns this data to the HTTP server (residing on the IT server called brandon). Through the use of HTML tags, the results of SQL statement execution are then displayed in the particular format chosen by the system developer.

One of the data structures used in the system for storage of Noticeboard information is as follows:

```c
struct
{
    VARCHAR catref[4];
    VARCHAR created[8];
    VARCHAR dat[8];
    VARCHAR title[50];
    VARCHAR description[100];
} in_rec;
```

A VARCHAR is an Oracle-supplied struct used for storing information in string format. This data structure allows the system to hold information on a particular Noticeboard entry in the database in respect of category reference number, date of creation of entry, creator of entry, title and associated general description.

### 3.4.3.2 Manipulating the Database using SQL

The struct discussed in the previous section has the capability of storing details on only one table entry at a time and so in order for the user of the system to view all available noticeboard entries an iteration process must be used which starts at the first table entry.
and works its way through the table until all records are retrieved. It does so using the following structure:

```c
for(;;)
{
    /*other code*/

    EXEC SQL FETCH cl INTO : in_rec;

    /*other code*/
}
```

the embedded SQL statement here ensures that all records in the table pointed to by the cursor variable cl will be stored, one at a time, into the `in_rec` struct for further processing.

### 3.4.3.3 Linking the Application to the Web

The application created was linked to the WWW using HTML forms as the user interface connected via CGI scripts to a database server machine. The Web forms allow for the user to input information or just browse information and these use the CGI script programs to process any information supplied. Any appropriate results are retrieved from the relational Oracle database used as the storage medium and these are then formatted by the CGI script so that they appear at the client side in a coherent fashion. The screen capture below shows the results of a Noticeboard search.
Figure 3.7

The TOPS Noticeboard Web Page
3.5 Object-Oriented Databases

3.5.1 Background

The principles of object-oriented programming originated in the 1960s. At this time programs were becoming increasingly large and complicated and almost impossible to reuse. Two schools of thought evolved to circumvent the problem. With structured programming the emphasis was on processing - the program is structured into individual subprograms. The other solution concentrated on data - also growing both in size and complexity so it needed to be grouped together according to shared characteristics. These two strands of thought were first brought together by Simula-66, which paved the way for modern object-oriented programming (OOP). The main aim of this programming paradigm is to create software that is flexible, extensible, reusable and easy to maintain.

In Simula the concept of a class, which groups together a data structure (object) and the procedures or methods for manipulating it, was available. Smalltalk descended directly from Simula and expanded on it by making a programming environment where everything is an object. Since then many languages were developed which provide object-oriented features, while preserving the traditional control structures of imperative languages (especially C), such as C++ and Objective-C. More recently, with the arrival of the internet and the need for distributed systems, the Java programming language has come to the fore. The features of such languages have been incorporated into the various OODBMS implementations which have emerged.
3.5.2 Data Model

It can be said that a truly object-oriented database is one which adapts the object-oriented features including encapsulation, multiple inheritance, support for abstract data types (ADTs) and function overloading to the context of databases. Such databases combine the elements of object orientation and OOPLs with database capabilities. They provide more than persistent storage of programming language objects and extend the functionality of OOPLs to provide full-featured database programming capability. Hence the data model for the application is well matched to that of the database. The ODMG-93 object model defines two types of so-called denotable objects: objects and literals. It further defines two types of characteristics of objects: operations and properties. Properties may be either attributes or relationships. Object databases can efficiently support any type of structure including trees and composite objects. A number of object-oriented database systems are currently in existence including Gemstone (based directly on Smalltalk and the first OODBMS to be developed into a commercial product in 1984), Orion (based on Lisp) and more recently ObjectStore (by Object Design Inc. and based on a C++ type interface with a fully capable Java version currently in progress) and Computer Associates Jasmine ODBMS product.  

3.5.3 Pros and Cons of the Approach

The advantages of object-oriented databases include
• it is easy to store the complex information (such as CAD diagrams and Multimedia Binary Large Objects) used in modern object-oriented systems since an ODBMS automatically understands objects.

• because object databases follow the persistent programming approach the program can manipulate transient data and persistent data in the same way using the same operations. Hence there is no impedance-mismatch problem as no data conversion is required. Also the programmer does not need to be aware of two different data models so less code is typically needed and maintainability is easier.

• every object in the system is automatically given an object identifier (OID) which is unique and immutable during the objects life. One object can contain an OID which logically references (points to) another object. This makes features such as bidirectional relationships easy to create.

• relationships between objects can be represented more efficiently by using a more convenient syntax than relational joins.

• the adoption of the Object Database Management Group's (ODMG) standard simplifies the process of making applications portable.

• many ODBMSs do not use a separate query language - the OOPL is the language for both the application and the database. This provides a direct relationship between the
application object and the stored object and the same operations can be used on both. The ODMG-93 standard also defines a declarative language called OQL for query of database objects.

- Most object databases support SQL for access to legacy relational database data.

3.5.4 Sample Object-Oriented Database Application

3.5.4.1 Modelling the Information

This sample application makes use of Object Design Inc.'s ObjectStore Persistent Storage Engine (PSE) for Java product. This implementation provides persistent storage for Java objects and is designed for applications which require persistent Java support for up to 100Mb of data accessed by a single user. The product has a small footprint (250Kb) and runs entirely within the Java application process.

The application itself sets up some information about people via a Person class, stores instances of the class in an object database and retrieves some of the data, displaying it to the screen. The workings of the application, Person.java, the code for which can be located in Appendix B is as follows:

- define the persistence-capable class Person. Each Person instance has a name (String), an age (int) and a set of children (an array of Person objects).
• define the main() method. This consists of three parts:

1. A call to ObjectStore.initialize(). This initialises the PSE software.

2. A call to createDatabase(). This creates a new database for read/write. A transaction for update is started and the three Person objects are instantiated within this transaction. Next a database root is created with the name “Tim” which is associated with the tim object and this root acts as an entry-point to the database called person.odb. Ending (committing) the transaction stores the person objects in the database.

3. A call to readDatabase(). This uses the “Tim” database root to access database objects which are finally used to print information to the screen.

3.5.4.2 Manipulating the Database

Database functionality is provided through the Java API which primarily provides functions to:

• create, open and close databases
• start and end transactions
• store and retrieve persistent objects

The following classes and methods provide the programmer's interface to the database functionality of ObjectStore PSE:
• class ObjectStore (to define system-level operations and parameters, initialising and shutting down ObjectStore)

• class Database (to open, close and destroy databases, creating and retrieving database roots\textsuperscript{13})

• class Segment (to provide physical clustering of objects)

• class Transaction (for starting, committing and aborting a transaction)

• class Persistent (defines storage for tracking an object's persistent status e.g. whether a persistent object has been fetched, dirtied, evicted and whether an object is actually persistent)

Once a class has been declared to be persistence-capable the standard Java constructs can be used to create and manipulate both persistent and transient instances of the class.

The Java API uses a postprocessor tool that analyzes the class files of persistent classes. The tool extracts schema information and inserts annotations that automatically fetch objects from the database when they are first accessed and automatically write objects back to the database when they have been updated. Transparent object persistence through ObjectStore and PSE enables the programmer to make use of the full power of Java and to easily integrate existing class libraries with ObjectStore or PSE.

\textsuperscript{13} Roots are used to label persistent objects that an application wants to retrieve by name.
All persistence-capable classes defined by applications (e.g. the *Person* class in this application) inherit from the abstract base class *Persistent*. There are four kinds of persistence-capable classes:

- String (e.g. *name* in this example)
- Arrays (e.g. *children* of type *Person* in this example)
- The Java primitive wrapper classes: *Boolean*, *Character*, *Double*, *Float*, *Integer* (e.g. *age* in this example), and *Long*
- Application defined subclasses of *Persistent* (e.g. *Person* in this example)

Once the program has been compiled using the java compiler, the *postprocessor* must be run on the resulting bytecode file - *Person.class*. This will make the *Person* class persistence-capable.

Now the program can be run as a normal Java application as follows:

```
java COM.odi.demo.people.Person person.odb
```

where *person.odb* is the application's database filename.

Running yields the following program output:

```
Tim is 35 and has 2 children named: Sophie Joseph
```

Each time the application is run it replaces the current *person.odb* database
3.5.4.3 Linking the Application to the Web

As in the relational example, there are many ways to connect this application to the Web. For instance, with slight coding modifications, the application could be set up as an applet to run in a Java-enabled Web browser such as Netscape. The applet might display some background information on what the application is meant to achieve and how it does so. It could also have a button which prompts for a user mouse-click in order to retrieve the PSE for Java database information dynamically and return the appropriate results to the browser. Alternatively the application could remain stand-alone and a button on a HTML page could run the application via CGI scripting. This would require some HTML "coding" within the Java application, however, in order to produce a browser response.

3.6 Object-Relational Databases

3.6.1 Background

A new market has recently been established in the database arena - the object-relational database sector - and by 2005 it is predicted that its market share will be 50% larger than the relational market [Stonebraker 96]. Two main causes can be attributed to this sudden domination; computerisation of new multimedia applications for the WWW and medical systems plus the addition of more complex functionality to existing relational DBMS applications. Object-relational databases are so-called as a result of their support for the SQL92 data model (relational) and for complex objects (object-oriented). Essentially they are a marriage of SQL from the relational world and the modeling primitives from the object world and serve the needs of relational users who can benefit from some object
extensions. Vendors of such DBMSs are increasing rapidly and include many of the major RDBMS vendors such as Oracle, Illustra/Informix, Omniscience, IBM and UniSQL. At present there is no standard for the object-relational model so it is not possible to clearly state what defines an object relational database product. Hence each vendor has their own object extensions to a greater or lesser degree [Bancilhon 96].

3.6.2 Data Model

ORDBMSs employ a data model that, according to the recent SQL3 standard, attempts to add "OO-ness" to tables. All the database information still resides in tables but some of the tabular entries can have a richer data structure, termed abstract data types (ADTs). Such a data type is constructed by combining basic alphanumeric data types. The support for ADTs is attractive because the operations and functions associated with the new data type can be used to index, store and retrieve records based on the content of the new (perhaps, multimedia) data type. Although the approach aids the modeling of more complex data, it lacks the fundamental object requirement of encapsulation of operations with data. It also has limited support for relationships, identity, inheritance, polymorphism, extensibility or creating user-defined persistent classes. In addition, integration with host languages like Java is not well supported so programmers are still forced to translate between objects and tables.
3.6.3 Pros and Cons of the Approach

The advantages of the object-relational approach include

- support for base data type extensions in an SQL context - SQL-92 restricts the available data types in a table column and defines a precise collection of functions and operators available for each type. This means that, for example, counting the number of zeroes in an integer is not an easy task since SQL's set of operations is limited. Extensible data types overcome this problem and increase application efficiency by allowing users to define new data types and the functions and operators that accompany them. For instance, consider the following hourly_pay function which divides an annual salary by 2000, the number of hours worked in the year. It can be set up as follows (using the ObjectSQL for Illustra ORDBMS)

```sql
create function hourly_pay (int)
returns float as
select ($1/2000);
```

The argument, $1, represents the annual salary of the employee. The function can now be used freely in any query, for instance,

```sql
select name
from emp
where hourly_pay(salary) > 12.50;
```

At runtime, an object-relational server substitutes the definition of the function into the query to produce the following query which is actually executed
• support for complex objects in an SQL context - this is achieved using three basic type constructors; composites (records), sets and references (pointers). Without these three mechanisms, there would be problems that could be very difficult or impossible to express.

• support for data inheritance from composite types and function inheritance which allows for the definition of new data types also. Multiple inheritance is also possible.

Data inheritance allows composite types to be grouped into an inheritance hierarchy which encourages modularity and consistent reuse of schema components. For example, consider the construction of a person_t data type with a single column, name:

```
create type person_t(name varchar(30));
```

Two additional data types can now be constructed, employee_t and student_t:

```
create type employee_t (salary int, startdate date, address varchar(30), city varchar(30), state varchar(30), zipcode int)
under person_t;
```
create type student_t (gpa float)
under person_t;

each subtype created, employee_t and student_t, inherits all of the data fields (name) from its supertype along with those defined within the subtypes.

The main problems associated with the approach are

• although the concept is relatively new, at present no so-called object-relational DBMS vendor implements all the object-oriented features required by a true object-relational DBMS. This has made it difficult to characterise the benefits of the technology and has the effect of making potential users suspicious. [Wells 98]

• Lack of support for integration with OO host languages is a major problem for ORDBMSs since it means information must be translated between tables and objects. This severely impacts the performance of the database.

### 3.6.4 Sample Object-Relational Database Application

#### 3.6.4.1 Modelling the Information

The Californian Department of Water Resources (DWR) [UCB 97] manages almost all waterways and canals in the state along with a collection of aqueducts and a massive state water project. To document their facilities, DWR maintains a library of 35mm slides
which has grown to about 500,000 slides. This library is accessed many times a day, mainly by employees. Typically, a user requests a picture by content but DWR has found that it is very difficult to find slides by this method. Indexing all slides according to a predefined collection of concepts is a prohibitively expensive job. Moreover, the concepts in which clients are interested change over time. DWR has written a caption about each slide, for instance, “picture of Auburn Dam taken during scaffold construction”. DWR also maintains a relatively primitive system that can identify slides from specified keywords. This keyword system does not work very well since many concepts of interest are not mentioned in the caption and are therefore difficult to retrieve.

Consequently, DWR is now scanning the entire slide collection into digital form and is constructing the following database:

```
create table slides (id int, date date, caption document, picture photo_CD_image);
```

```
create table landmarks(name varchar(30), location point);
```

Each slide has an identifier, the date it was taken, a caption and the digitised bits in Kodak Photo-CD format. This format constitutes a collection of five images ranging from
Chapter 3 - Storing Multimedia Objects for use on the Web

a 128 byte X 192 byte thumbnail to the full 2kB X 3kB colour image. DWR has digitized about 40000 images and the eventual database will be about 3TB in size.

DWR wishes to classify their images electronically rather than manually. One of the attributes DWR wishes to capture is the geographic location of each slide. It will achieve this through a public spatial database from the US Geological Survey which includes the names of all landmarks that appear on any topographic map of California, along with the location of the landmark. This is the table landmarks shown above. They propose to examine the caption for each slide to determine whether it contains the name of a landmark. If it does, the location of the landmark is a good guess for the geographic position of the slide.

3.6.4.2 Manipulating the Database using ObjectSQL

DWR is interested in writing image-understanding programs that will inspect an image and ascertain attributes of the image in the database. For example, it is possible to find a sunset in this particular slide library by looking for orange at the top of the picture. Low water in a reservoir entails looking for a blue object surrounded by a brown ring. Many attributes of a picture in which DWR has an interest can be found using fairly mundane pattern-matching techniques. Some attributes are much more difficult to ascertain, for example, to determine whether a picture contains an endangered species. These will have to wait for further advances in pattern recognition.
The clients of DWR’s database will submit ad hoc inquiries. One such inquiry might be to find a sunset picture taken within 20 miles of Sacramento. Clients need a friendly user-interface that assists them in stating the following SQL query:

```
select id
from slides P, landmarks L, landmarks S
where sunset (P.picture) and
contains (P.caption,L.name) and
L.location || S.location and
S.name = 'Sacramento'
```

The steps involved in making the query include

1. Find the geographic location of Sacramento (S.location) in the landmarks table.
2. Then find other landmarks (L.location) that are within 20 miles of Sacramento. || is a user-defined operator defined for two operands, each of type point, that returns true if the two points are within 20 miles of each other. This is the set of landmarks that can be used to ascertain if any appear in the caption of a picture.
3. The user-defined function contains accepts two arguments, a document and a keyword, and determines whether the keyword appears in the document. The function contains yields the set of pictures that are candidates for the result of the query.
4. Finally, sunset is also a user-defined function which examines the bits in an image to see whether they have orange at the top. The net result of the query is the one desired by the client.
### 3.6.4.3 Linking the Application to the Web

At present there is a collection of over 15,000 photos from the Californian DWR online as part of the testbed for the University of Berkeley Digital Library Project. The system uses the Illustra/Informix object-relational DBMS for managing the images and their associated textual data. It uses a cgi-bin program to take information from Web forms and construct an SQL query to the database. The user-interface looks as follows:

<table>
<thead>
<tr>
<th>Search</th>
<th>Clear Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Photos to Display Per Page</td>
<td></td>
</tr>
<tr>
<td>Photo Caption</td>
<td>a word or phrase that describes the picture</td>
</tr>
<tr>
<td>Year of Photo</td>
<td></td>
</tr>
<tr>
<td>Subject</td>
<td>See a Complete Subject List</td>
</tr>
<tr>
<td>Category</td>
<td>Location</td>
</tr>
<tr>
<td>Horizon</td>
<td>Yes  No</td>
</tr>
</tbody>
</table>

-153-
I filled in the above Web form with the following information:

<table>
<thead>
<tr>
<th><strong>Photo caption:</strong></th>
<th>Water</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Year of Photo:</strong></td>
<td>Omitted</td>
</tr>
<tr>
<td><strong>Subject:</strong></td>
<td>I chose <em>animals and wildlife</em> from the list</td>
</tr>
<tr>
<td><strong>Category and Location:</strong></td>
<td>I chose the default <em>any</em> from the lists in both cases</td>
</tr>
<tr>
<td><strong>Horizon:</strong></td>
<td>I chose <em>no</em></td>
</tr>
<tr>
<td><strong>Perspective:</strong></td>
<td>I chose <em>ground</em></td>
</tr>
</tbody>
</table>

and within seconds received 3 JPEG images with the following captions:

- "Wild ducks and ducklings in water"
- "Ducks in water"
- "Close-up of a male mallard duck in water"

and other traditional relational information such as subject and ID data.

Table 3.1 below compares the four types of storage mechanisms discussed from a number of perspectives in relation to their support for multimedia information and Web applications.
### Table 3.1

<table>
<thead>
<tr>
<th>Storage Type</th>
<th>Performance</th>
<th>Access</th>
<th>Query</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>File System</td>
<td>Good for simple static Web pages - access is fast</td>
<td>access is from the file system directly - no query language</td>
<td>Not possible</td>
<td>Simple, info held in file system directories</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relational</td>
<td>Good for Web apps using basic data types - not so for multimedia data where info stored in a BLOB data type</td>
<td>Access to multimedia info is through SQL query language</td>
<td>Not possible to query contents of a BLOB</td>
<td>Info held as a series of bytes in a BLOB</td>
</tr>
<tr>
<td>Object-Relational</td>
<td>Not good since data must be continuously translated between tables and objects</td>
<td>Access is through a proprietary Object-SQL type query language</td>
<td>Possible to query multimedia data using pattern matching techniques</td>
<td>Multimedia info is mapped to rows in a table</td>
</tr>
<tr>
<td>Object-Oriented</td>
<td>Fast since no translation is necessary - the database can actually store complete objects</td>
<td>Object Query Language with support for SQL</td>
<td>Possible to query multimedia data using pattern matching techniques</td>
<td>Multimedia information is stored as complete objects in the database</td>
</tr>
</tbody>
</table>
3.7 Conclusions

The techniques by which information used on the Web can be stored are many and varied. Organisations wishing to set up a presence on the Web need to understand fully the requirements of their application, the complexity of the information involved and the storage options available to them.

The file system storage mechanism has its roots in early data processing times and is still used today by several systems. It is severely limited in the sense that any commercial applications which use it for storage purposes are highly inflexible and data dependent. In relation to Web information storage, it is most effective as a container for non-application specific multimedia data to be used in the Web site. This includes icons, logos and general decoration data such as animated images and applets which can be retrieved most efficiently via the file management system.

The concept of the file was replaced by that of the relational database in the early 1970s and today the majority of commercial applications still use this storage mechanism. It remains the most popular storage technique with the lion's share of the marketplace. It has been so successful to date mainly due to its intuitive data model (tables) and the fact that traditional applications have only needed to use fundamental data types such as strings and characters. Many relational databases can now store multimedia data using the BLOB
data type but such information cannot be queried and, in any case, only a limited amount of data can be stored.

With the requirement for organisations to use more complex data types, relational database vendors are now adding object extensions to their products to give their customers some of the functionality they require. These products are called object-relational database management systems and combine both the relational and object approaches in one system. The major drawback to the approach is that much processing needs to take place in order to translate objects into a relational table and vice versa, degrading performance. In addition no standard exists for such DBMSs so vendors are advertising products as object-relational when they should really be called extended-relational. The release of the SQL3/4 standard next year should help enormously in this respect. From the Web storage aspect, object relational applications have been developed which store vast quantities of multimedia information, so it is an appropriate solution for organisations who wish to remain tied to the concept of relational tables but who need to store complex data types. Many experts in the field believe that, although the object-relational approach is not appropriate for the storage of complex data due to its disparate data models, it will catch on because the ORDBMS vendors are so powerful in the database sector [McClure 97].

The object database storage mechanism is a truly object-oriented solution to database management. An ODBMS stores objects as objects, no mapping or translation is required. ODBMSs tie an object oriented programming language to the database and use it to
create, store, update and retrieve persistent objects from the database. This is very significant as it means that a query language is not essential to carry out database queries and manipulation, everything can be done from within the host language. Transient and persistent objects can be treated in the same way, and operated on in the same way. ODBMSs also carry all the OO features associated with their host language which means creating user-defined classes and objects is easy. From a Web storage perspective, object databases can handle both traditional and complex data with relative ease and the next chapter of the thesis will illustrate this clearly. The fact that the data model of the database is closely tied to that of the host language means that storage and retrieval of such information is highly efficient.

Organisations will make choices of database vendors based on many criteria, including simplicity of use, simplicity of development, support for OOP, extensibility, complex data relationships, product maturity and support for legacy data (SQL). It is clear that in today's complex, rapidly changing world, ODBMSs provide the most flexible, extensible alternative for companies that need to match the capabilities of their information systems with the needs of their organisation. They are also tried and tested and have been used successfully in many commercial Web applications.
CHAPTER FOUR

The Analysis, Design and Implementation of a Multimedia Object Database Application for the WWW

"Putting object extensions on RDBMSs is tantamount to adding stereo radios and global navigation systems to horse-drawn carriages. You will have interesting enhancements but the wrong base vehicle."

Steve McClure, Director of Internet Tools, International Data Corporation

This chapter discusses the steps involved in the creation of an application which utilises object database technology to store multimedia data along with other complex information types. Initially, the object-oriented features particular to an object database are reviewed. Next, the vehicle upon which the application is created is investigated as to its suitability. The analysis, design and implementation aspects of the system are then illustrated in detail and finally the application is tested.
4.1 On-line Interactive Processing and Object Databases

We are currently in the midst of an electronic commerce (e-commerce) revolution brought on by the internet. Electronic commerce can best be defined as “customer self-service” applications made user-friendly by introducing rich multimedia content in a secure processing environment. As customers are interactively navigating through a wide variety of multimedia data and text, such applications are termed “On-line Interactive Processing” (OLIP) [Finkelstein 97]. Organisations are deploying e-commerce OLIP applications on the internet as a means of creating a customer self-service environment. Such systems extend an organisation's customer base and provides more responsive and cost effective service by eliminating the middle person (so-called disintermediation). Many such OLIP systems involve the user in browsing through a catalogue of products (with associated images and textual information) and in such cases, a back-end database is essential. Furthermore, it may be concluded from the previous chapter that the most suitable technology to use for storing such complex data is an object database.

4.2 System Analysis

4.2.1 Performing OOA on the System

4.2.1.1 System Requirements

The system which is to be created involves two main components:
• an object database

• an application which involves performing some actions on the database

From the application viewpoint it is necessary to decide what the objects to be used in the system are. In this way it will be possible to create a database which can store and retrieve instances of such objects. What is required at this stage is a general specification of the classes which are needed to make the system function properly.

The application to be developed is intended to serve as an electronic educational tool. It attempts to guide the end-user through the various facilities which can be used with an ODBMS called Jasmine (this is actually the implementation tool used to create the application itself and will be dealt with in more detail later). The application involves several component parts, each with its own unique functionality, as follows:

• *Log in to system* - when the end-user runs the application, this is the first component ‘screen’ to be presented. Its main functionality involves taking a user/password combination from a prospective client and allowing that client to view the remainder of the application only if their account details comply with those already stored in the database. If not an appropriate message will be output. If the client does not have an account in the database or has forgotten their account details they can also contact the system administrator from this screen. This action will take the user to a different component screen.
• **System Administrator** - the user will be presented with this screen if they require account details of some description. A message is shown indicating to the user that they can contact the system administrator using a number of modes such as phone, letter, e-mail etc. However, the user can only visualise the required information from the database by clicking their mouse on the appropriate button.

• **Home Scene** - after the user has connected successfully to the system this is the screen they will be presented with. It should welcome the user to the application, present some information on the use of widgets and their associated behaviour, and give a list of several further screens which the user may inspect.

• **Movies and Audio** - in this component screen the user should be given a verbal description relating to the audio/visual multimedia capabilities of the ODBMS. The user should be given the opportunity to initiate a movie and an audio clip stored in the database by simple mouse clicks.

• **ODQL Interpreter** - this component screen should attempt to give the user some understanding of Jasmines Object Database Query Language (ODQL) Interpreter tool. The user should be given functionality which facilitates them in launching the interpreter environment and some sample ODQL code is provided to enable the user to query the database first-hand. The user should also be able to check some of the output on the
ODQL monitor against the information currently in the database via a mouse click. In order to launch the interpreter, its executable must be present on the clients machine.

- **ActiveX Controls** - this screen should outline the support which the ODBMS has in relation to the ActiveX Control and Code Component. It should allow the user to play a video by clicking on a video ActiveX control. In addition it should display an OLE object and allow the user to open it using its associated application.

- **Interactive Transaction** - this component of the application should allow the user to carry out a specific update transaction on the database interactively, using only button clicks. It should allow the user to see the end-result of their transaction, and allow the user to commit or abort the transaction.

- **OLIP Component** - this part of the application should contain a 'sub-system' which allows the end-user to browse through the thumbnail images of a fashion catalogue. If the user requires a closer look at any particular item in the catalogue, a larger image should be displayed and the associated textual details (price etc.) retrieved from the database.

- **Web Browser** - this application segment should indicate the ability of the Jasmine ODBMS to allow users to 'surf' the WWW via its Web Browser ActiveX control. Other functionality should be added to allow the end-user to navigate the Web easily. In particular, a button click should allow the end-user access to the ITT Intranet over the internet.
As a result of the preceding system requirements it is now possible to state the objects required for the overall application as follows:

- system account (or customer) objects
- widget functionality objects such as buttons, list boxes, static text boxes etc.
- multimedia objects such as images, sound and video
- garment objects for the fashion subsystem
- ActiveX control functionality objects

4.3 System Implementation

4.3.1 Introduction

With most applications, once the design stage is complete, it is possible to implement it using a number of modes. However, in this case it is a "chicken and egg" scenario. The application attempts to discuss the capabilities of the Jasmine ODBMS. Hence, familiarity with the implementation tool itself was actually necessary before analysis began in order to get a feel for what could actually be designed and implemented. In this way it was possible to come up with a set of system requirements. The following subsections will detail the implementation of the design.
4.3.2 Implementation tool - Jasmine

The key feature common to all these new Web applications is multimedia, which, when used in conjunction with traditional data elements, brings applications to life and makes them much easier to use. The fact that these applications are dynamic means that they must be built upon a solid database foundation. Such databases must now manage and deliver complex data to a large number of users.

4.3.2.1 Features of the Jasmine ODBMS

These include [CAI 97a]

- modelling complex information

- combined data access and OOP

- Object Database Query Language (ODQL), standalone or embedded within C/C++.

- Interfaces with C/C++

- Multimedia and ActiveX structured storage support.
• ActiveX support - the Jasmine ActiveX custom control and code component open the Jasmine database to systems like Visual Basic, which support these standard protocols.

• Distributed client/server architecture where a single server can support several databases and each client can access multiple databases on many servers.

• can execute as a standalone client application or as a plug-in to a Web browser.

• Database integration with other databases, such as OpenIngres, Oracle and Informix RDBMSs.

• Data management services including locking, concurrency control, transaction management and recovery.

4.3.2.2 Jasmine Design and Development Facilities

• ODQL Facilities - ODQL is a complete database programming language that provides a number of general purpose programming facilities such as constructs to declare variables, assign values to variables and use variables in expressions. In addition it provides an iteration facility. It can be used in application programs written using the Jasmine C API but also in methods used to implement database operations. It is available in three environments; as an interactive tool with the ODQL interpreter (very useful for testing
ideas), as a method creating tool with the ODQL pre-processor (which also allows the construction of dynamic ODQL statements at run-time) and finally in conjunction with the C API which involves the execution of ODQL statements and commands via a C API function.

- **Class Libraries** - Jasmine provides a set of system classes for the basic data types. Native support is also supplied for various kinds of collections and for system management such as transaction and session control. The system class hierarchy also provides the CAComposite class which is used as the base class for all user-defined classes. The ODBMS comes packaged with several class libraries, including a full-featured multimedia library for creating, storing and retrieving multimedia information.

- **ActiveX Control and Code Component** - these allow systems such as VB, which support ActiveX, to access Jasmine database information by executing queries in the Jasmine ActiveX control.

- **Weblink** - this tool provides access to the Jasmine database across the internet (or intranet) using HTML pages.

- **Jasmine Application Development Environment (JADE)** - this is a Windows application with an easy to use GUI that allows for the creation of Jasmine applications. It
is the main environment used to create the database application for this thesis and will be discussed in more depth presently.

### 4.3.2.3 Jasmine ODBMS Concepts

A number of OO concepts have already been discussed at the outset of this chapter and these are all applicable to the Jasmine ODBMS, as well as other ODBMSs such as ObjectStore. These might be called 'common' OO features of modern ODBMSs. What is discussed now are the concepts particular to the Jasmine ODBMS.

**Stores** - the containers in which data is held within Jasmine. Four types exist, the most important of these from a development viewpoint being the *system store* (created automatically when Jasmine is installed) used to store Jasmine system classes and the *user store* for storing user objects. A number of user stores normally exist, set up by the system administrator.

**Class Families** - this consists of a collection of related classes. They are often reusable and can be written to be employed in a wide range of applications. A store can hold several class families. The classes within a class family are arranged into a hierarchical relationship forming a tree within the overall class hierarchy. Importantly, a class cannot inherit from a class in a different *user-defined* class family. However, relationships may cross the boundaries between class families. All class families have the system class Composite as their top class.
Properties - In Jasmine, information held about instances of a class is defined as a set of properties. A property may store a single item of information or a collection of items of information of the same type. They can be *instance-level* properties where a different value generally exists for each class instance (the norm) or *class-level* properties where a value applies to a class as a whole. Properties can be of two types - *attributes* which are items of raw data (atomic literals or a collection of these) or *relationships* (references to another entity (class object) or collection of entities).

Properties of a class can be *mandatory* where they must always be given a value. They may also be given default values if none is given upon creation. A property which is *not* mandatory may be NIL. This is not a value (zero) but rather the state of a property that currently has no value. Similarly, for collection-valued properties, being empty is not the same as being NIL.

Unique properties are also possible where a constraint of uniqueness can be applied to both new and existing single-valued class properties.

4.3.3 *Jasmine Application Development Environment (JADE)*

The various modes by which a Jasmine ODBMS system can be created and modified were mentioned earlier. From the market viewpoint, there is no faster way to implement a design than by the use of visual tools since it abstracts most of the underlying (C/C++)
code on which it is based. Hence, this was the preferred approach for the educational system being developed. In addition, there is the effect of "getting two for the price of one" in that once an application has been developed using JADE, it is possible to run the same application as a stand-alone client/server or as a Web-enabled application. Both executable environments are discussed in this thesis, with focus on the Web-enabled version in the next chapter.

4.3.3.1 The JADE Development Process

The steps involved include

- Locate or create items in a Jasmine database for the application
- create application scenes
- create and/or customise database objects
- test and debug the application
- compile the application
- build the client/server executable file or publish the Web-enabled application.
- deliver the following to the end-user:
- a disk set or CD that contains the runtime install set, the application executable file, and the Jasmine client.

- the Netscape or Internet Explorer Web Browser install set and the applications Web URL.

### 4.3.3.2 The JADE Environment - Application Manager

This is the GUI (see Figure 4.1) used to select and manage JADE applications and their scenes (the "pages" of an application for designing its visual run-time display), debug, run and compile scenes/applications and create distribution executable files for client/server applications or publish HTML files for Web-enabled applications. It also allows you to display the JADE class browser to administrate the database.

**Figure 4.1**

*Jasmine's JADE Application Manager*
4.3.3.3 The JADE Environment - Class Browser

The class browser (see Figure 4.2) may be selected from the application manager window where the user can work with all the items (class families, classes, objects, queries, methods) in the Jasmine database and local widgets built-in to JADE. It is possible from this interface to drag local widgets, classes, objects and queries onto a scene or display other inspector and editor windows to create or edit class and object properties, construct queries or program methods. This is the tool which is used to create the Customer and Garment (sub)classes for the application.

Figure 4.2

The JADE Class Browser
4.3.3.4 Creating a New Application in JADE

An application object must exist before any scenes can be added to it. Once the JADE executable is launched, the application manager window opens and from here it is possible to create a new application object. The user must first connect to the Jasmine database manually however using the Jasmine Connections dialog box. For stand-alone client/server applications a built-in LOCAL connection exists and this can be used to connect to the database. Alternatively the user may create their own connection to the database (and this is mandatory for Web-enabled applications as seen in the next chapter).

Once the connection has been made a new application can be created easily by clicking the New Application toolbar button. The user only needs to enter a name for the application in the New Application dialog which appears. The only stipulation is that application names must be unique. In this case the application is called MScApp.

From the application manager window it is now possible to create individual scenes

4.3.3.5 Creating a New Scene in Jade

Once the new application object has been created, it is possible to create new scenes for the application by selecting the applications name in the class browser window and clicking the New Scene toolbar button. As for applications, each scene name must be unique. When the user selects a particular scene name from the class browser window,
the scene window opens in edit mode. This means it is now possible to add/delete/modify any items which exist in the Jasmine database to the scene.

4.3.3.6 Adding Scene Items and Defining Item Behaviour

The functionality of the application has already been overviewed so now the classes, objects, queries, widgets and methods to be used with each scene must be quantified. These requirements are as follows for the Log In scene.

- four static text boxes (Local Widget CF), for welcome message, secondary message, login message and system administrator message

- image display placeholder (Media CF) for ITT Crest

- two edit boxes (Local Widget CF), for inputting the username and password

- three buttons (Local Widget CF), for submitting the login data, accessing the SystemAdmin scene, and exiting the system

- default background, for setting the scene background (this object is part of every scene automatically and cannot be removed).
These are the "visual" components of the scene - those that will be seen at run-time. However, in order for the scene to perform its duties, some "invisible" components need to be added to the scene also as follows:

- Customer class object to house a collection of the Customer names and passwords in the database.

- Query object (of the Customer class, CAStore CF) so that when the user submits a name/password combination, these will be compared against those in the Customer class object. If there is compliance, the user is brought to the Home scene, otherwise an error message box is returned to the screen.

All the objects discussed above are added to the JADE scene by the "drag and drop" method from the appropriate inspector or editor window. In addition at this stage all the objects are standalone entities in the sense that they will perform no useful function on their own. Behaviour must be defined for each object so that it can interact with either the user directly at run-time or with itself or other scene objects. For example, for the values which the user inputs for name and password to become "common knowledge" to all the scene items the following behaviour must be defined for the associated edit boxes:
Essentially this sets the edit boxes information up as global variables to the program so when the user then clicks the login button a message is passed to the "invisible" query object selected in Figure 4.4 in the top right of the JADE Scene's Edit window.
The query object's behaviour is by far the most complicated of the scene items in this case. When it receives the message from the login button it does two things. First it executes its query using the global variables just declared as parameters. This query is defined via the Query Editor window as shown in Figure 4.5 below.
So, after the query is performed the query object should hold either a single Customer object or no Customer object (NIL) if the name/password was incorrect.

It then performs a conditional message in relation to these two outcomes as shown in Figure 4.6
Essentially, the first condition relates to the possibility of a NIL value for the query object. Here, a further message is sent to the query object itself. Its purpose is to display a message box on screen saying “You have entered an invalid customer name or password - Please try again”.

The second condition relates to the username/password combination actually existing in the object database or if not, adding the new Customer details interactively. In this case a message is sent back to the query object itself. Its purpose is to set up another global variable (called login) and populate an "invisible" Customer class object with this value.
This essentially paves the way for adding new customers to the Customer class interactively - although this functionality itself is not used within the system. After this, a timer is started and connection is made to the Home scene from which any other application scene can be accessed.

The LogOn scene looks as shown in Figure 4.7 at run-time. It is important to note that there are a host of message and actions types defined for the various Jasmine objects [CAI 97b]. Understanding these is essential in order to create an application which will perform as required. However, their discussion is outside the scope of this thesis. In addition, a discussion of the ODQL interpreter [CAI 97c] and its associated commands and statements along with Jasmines ActiveX control and code component[CAI 97d] cannot be discussed here for brevity. However, both need to be understood in order to utilise their functionality.
4.3.3.7 Testing, Saving, Debugging and Compiling a Scene

While a JADE scene is being created it can be tested by running it. This can be done from the application manager window. The scene may be stopped from the same window. This automatically returns the scene to edit mode. A scene may also be saved (committed to
the Jasmine ODBMS) from the application manager. Running the debugger on a scene allows the user to step through the scene and view any errors which appear. Compiling a scene compresses and places scene data in logical order.

### 4.3.3.8 Creating a Distribution Executable

When the application has been created and fully tested, it is possible to create a standalone client/server application. This may be done from the application manager window by selecting the application name, in this case MScApp, and then selecting the **Create Distribution Executable** command from a pull-down menu. A warning message is displayed, prompting for the applications scenes to be compiled. Once this is performed successfully JADE presents a dialog which allows the user to select an icon for the executable, an executable name and a default connection name to start the application, **LOCAL** in this case. Once this information is supplied JADE creates the distribution executable file automatically. The executable can then be run from the local machine provided that the Jasmine client is resident. The JadePlayer executable runs all JADE client/server applications. The Audio/Visual scene is illustrated in Figure 4.8 below running as a distribution executable. Notice that the lady with the blue dress is turned, this is because the scene was captured while the movie was playing. All the static application scenes are located in Appendix C at the end of the thesis, captured while running standalone.
When most people think of multimedia they usually expect some combination of sounds and sights. Traditionally, database storage techniques have been rather limiting in the sense that the data stored was restricted to textual information such as strings and numbers. The Jasmine object database model overcomes these restrictions and stores images and sounds just as easily as relational database type data. I have set this scene up so that if you left-click on the ITT crest above, it will activate an .avi movie related to the static GIF image on the left. It will appear that the static image has disappeared but this is not true. It is simply hidden from view while the movie plays and will reappear once the movie is over.

You will probably notice that this movie is silent. If you want to get a taste for the audio capabilities of this object database technology, just left-click on the image of the cartoon character, Homer Simpson, on the left and hear him "sing"! Afterwards you may regret listening!

The need for system analysis and design is a major factor in today’s applications, which are becoming increasingly complex. Essentially, proper analysis and design means far less post-development software maintenance. There are many different analysis and design techniques in common use but for object-oriented applications, such as the ODBMS application in this case, OOA/OOD techniques should be used if possible. The
OMG have recently ratified a standard notation for OOD called Unified Modelling Language (UML) and it appears that it will become the notation of choice for the vast majority of organisations.

With the requirements for multimedia and complex data types in today’s business applications, there is huge pressure on organisations to keep ahead of their competition. This is a time-to-market issue in many cases. Computer Associates, an American software giant, have created the Jasmine ODBMS as a result of these demands. As a development tool it enables the creation of a diversity of applications using a variety of platforms. This author chose the Jasmine Application Development Environment (JADE) visual tool to implement an electronic educational application. JADE itself is implemented using the Jasmine C API but, apart from creating methods, the designer sees very little code. It enables the creation of multimedia applications in very short timeframes and is welcome news from a corporate perspective. It utilises a drag and drop approach to application design and integrates with standards such as ActiveX and OLE. In addition it is possible to integrate Jasmine with many relational databases using its SQL capabilities. Applications created in JADE consist of one or more scenes, which in turn contains one or more scene items. A scene item reflects anything that can be stored in the Jasmine database such as classes, objects and queries. Scene items are dragged onto the JADE scene from appropriate windows and in order to give them functionality, behaviour can be defined. The testing, debugging and compiling of JADE applications is very easy and distribution executables can be produced using JADE very quickly. This author believes that Jasmine will become the vehicle of choice for many organisations
worldwide who require multimedia and complex data in their back-end databases quickly. CAI have already received widespread acclaim for the product from industry analysts and are in partnership with a number of organisations such as Excalibur, whose RetrievalWare product may be integrated with Jasmine to perform queries on text documents and image archives stored in the Jasmine ODBMS [Excalibur 97]. It is the only ODBMS of its kind available today and it remains to be seen how long it takes before the competition modify their own ODBMSs.
CHAPTER FIVE

Running an ODBMS Application on the WWW

"Using tables to store objects is like driving your car home and then disassembling it to put it into the garage. It can be assembled again in the morning, but one eventually asks whether this is the most efficient way to park a car."

Esther Dyson, Database industry expert

This penultimate chapter discusses how the electronic educational system developed in the last chapter can be published on the WWW, making it available to a global audience. Testing in relation to database performance factors over the internet is carried out and analysed from a number of perspectives.
5.1 Publishing the Jade Application on the WWW

To publish the educational system on the WWW requires a number of steps, each of which is detailed in the following subsections.

5.1.1 Creating a “Local” Database Connection

For distribution executables it is possible to use the LOCAL connection which comes with the Jasmine installation to connect to the database. However, this is not possible for publishing Web-enabled applications. Here a user-defined “local” connection can be created containing a username, password, network IP address, network protocol and Jasmine server listen address. Username and password in this case reflect the login used to enter the machine upon booting rather than the system database username/password combination. The new connection is created in the JADE application manager window by selecting the Database Connect button on the toolbar. The new connection details can be entered and edited using the window which appears and connection to the database can be tested also (see Figure 5.1)
5.1.2 Installing a WWW Service

The application cannot be published on the Web until a WWW service has been installed. In this case, Microsoft’s Internet Information Server 2.0 was installed as the Web server software [Microsoft 96]. The main reason for this was because the Jasmine developer edition being used required the Microsoft Windows NT 4.0 platform as did the Internet Information Server software. Hence, it was felt that there would be better integration
between all parties involved if Internet Information Server were used rather than another
Web server product such as Apache HTTPd. The main result of the installation is that a
directory called wwwroot now exists where HTML files can be placed for global access
on the Web.

5.1.3 Installing a JADE Web Browser Plug-in

The successful publishing of a JADE application for the Web creates a file whose
extension is .sdj and whose MIME type is given by application/x-sdj. For this
to be translated for viewing on the Web, the JADE Web browser plug-in is required. This
needed to be installed from the Jasmine CD for both Netscape Navigator and Internet
Explorer browsers (a different plug-in for each browser) and then verified.

5.1.4 Configuring System Parameters

In addition to all the requisite installations, a number of intricate system configurations
needed to be deployed. These details were not available in any manuals which this author
read and so technical support was requested directly from CAI in New York. The main
thrust of these configurations dealt with user rights for anyone attempting to access the
application over the internet. The most important of these was to give user rights to
access this (the machine on which the Jasmine server resides) computer from
the network. Additionally an alias had to be set up in the Internet service manager for
the C:\Jasmine\HTTPServer directory called /Jade. A few other minor
modifications were also required in order to get the application running smoothly on the Web.

5.1.5 Publishing the Application on the Web

Now that all the component parts are installed it is now possible to publish the application. This was done by selecting the application name, MScApp, from the class browser window and then selecting the scene called LogOn from the associated list of scenes. Doing this tells the publishing executable that the resulting LogOn.html will be the starting scene (home page) for the Web application. From the Tools menu in the application manager, Publish Web Application can then be selected. The user is then given the option to compile the scenes of the application and then a Publish Web Application dialogue appears. This window contains information relating to the locations of the HTML file and connection file which will be produced upon successful publishing of the application. The default locations displayed may be altered but it is essential that the Connection file reside in a directory beneath C:\Jasmine\HTTPserver and that the HTML file reside in a directory beneath C:\Inetpub\WWWroot. For the Use Connection edit box the user should enter the connection to be used, in this case it is JBConnect, a “local” connection set up earlier. Once all the details have been entered the OK button on the window may be clicked to publish the application. If the application publishes successfully the Web-enabled application will reside on the Jasmine server and the corresponding connection and HTML files copied to the specified locations on the Web server.
5.1.6 Running the Web Application

In order to run a JADE Web-enabled application three servers need to be operating. These are the WWW server, the Jasmine database server and the Jasmine HTTP server. If any one of these is not operating the application cannot run in the browser. Hence the first step involves verifying that all three servers are up and running. This can be done by checking the server manager under admin tools. If any of the servers is not operating it should be started manually.

Once these servers are operating successfully it is then a simple matter of launching the Web browser (both Navigator and Explorer were tested) and directing its address box to the following URL:

http://193.1.182.257/LogOn.html

Once connection has been made to the application, the client retrieves the starting scene files from the Jasmine server and these are then placed in the Temp directory of the client machine. When the user selects any other scene the same process occurs. It is worth mentioning that multimedia information such as images and sound clips are retrieved dynamically from the database when the scene where they reside is accessed. Of course, if a scene is revisited after retrieval from the server, it will be accessed much more quickly as it is already in the client cache. Figure 5.2 illustrates the MoviesVideo scene running in the Netscape Navigator browser.
Chapter 5 - Running an ODBMS Application on the WWW

Figure 5.2

The Educational System's MoviesVideo Scene Running as a Web-enabled Application

5.2 Investigation of Database Performance Factors on the Web

5.2.1 Test Methodology

The purpose of this investigation is to discern the single-user performance of the object database on both a local and remote machine over the internet. For comparison, Netscape Navigator is used for both machines to run the Web-enabled educational system. For all
tests performed, the data retrieved from the database is "raw" in the sense that before each test the following occurs:

- the Jasmine database server is stopped if running and restarted
- the Jasmine HTTP server is stopped if running and restarted
- the contents of the Netscape Navigator cache is cleared
- the contents of the Netscape Navigator history folder (which holds any previously accessed URLs) is cleared

This ensures a consistent starting point in every case so that nothing is previously stored in any cache, either client or server.

The tests performed are as follows:

- **Cold vs. Warm time** - Cold time is the time taken for the first request, in this case the time taken to retrieve the LogOn scene from the database. This first request causes the database to be opened, the data to be paged in, and the pointers to be 'swizzled' [Varela 95]. Warm time is the time taken to perform subsequent requests. In subsequent cases, the database is already opened so access times should be faster. This test is performed to measure the savings obtained by having the intermediate Jasmine HTTP data server (which keeps the database open across multiple requests).
Chapter 5 - Running an ODBMS Application on the WWW

- Remote vs. Local request - the "remote" machine in this case is a machine physically remote from the server machine (even though it is in the same research lab). Both local and remote machines are hooked up to a fast 10mps line and so can be compared to observe the effect of the network on the access times.

- Large vs. Small result - this test is performed to observe the effect of the query result size on the times.

- Morning vs. Evening - this test is used to check the effect of network traffic and machine load on the times.

5.2.2 Analysis of Test Results

The following subsections detail the data access times encountered for the scenes in the online educational application from both a local and remote perspective.
5.2.2.1 Cold vs. Warm Times

It can be seen clearly from the graph of average scene access times that the cold times i.e. the times to connect with the Jasmine server and access the LogOn scene are far greater than any warm times, as expected. This is because for warm time access, the database has already been opened and so a new opening is not required, saving much needed time. The effect is consistent for all cases, local and remote, morning and evening.

5.2.2.2 Small vs. Large Result

The average scene access time graph clearly indicates that scene access time is a function of the associated scene complexity. For instance, the Catalog scene is certainly the most demanding of the individual scenes in that it involves populating three display placeholders with bitmap images, each of which must be taken from the database and transferred to the client. This is reflected in the graph above for both local and remote
cases. In a similar vein, the ODQl and Web scenes are the most basic of all in that they involve no transfer of multimedia data from the database and contain mainly static text boxes and buttons. Quite clearly, their relative simplicity is shown by their associated access times.

5.2.2.3 Morning vs. Evening Request

![Remote Machine Performance](image)

![Local Machine Performance](image)

-197-
The graphs above show that for both the local and remote cases, the morning access times are typically a little slower than those for the evening. This is as expected since the network experiences much less traffic after six o' clock when day-time classes have finished compared with the morning period when many staff and students are using the local network.

5.2.2.4 Local vs. Remote Request
The graphs above indicate that for both morning and evening requests to the database, remote access is typically between 10 and 20% slower than for the local case. This is to be expected, considering the effect of the network and the inevitable extra routing of information needed for remote access to occur.
"Any sufficiently advanced technology is indistinguishable from magic."

Arthur C. Clarke, author of 2001: A Space Odyssey

The creation of WWW servers which accommodate multimedia information have become of paramount importance in recent times due in no small way to the emergence of online e-commerce and the general need to design Web sites which will visually appeal to the end-user. The design of an efficient, workable system can be a difficult and complex task. The functionality of any system is likely to be application dependent so there is no set of "hard and fast" rules which solve every foreseeable problem. This chapter considers the most likely future directions which WWW server development will take and the conclusions to be drawn from this work are outlined and discussed.
6.1 Dynamic WWW Server Development

All the signs and surveys would seem to indicate that the WWW is here to stay and remain popular for the medium to long term. There is no doubt in this author's view that this will be the case since the 'Internet' has captured the imagination of millions worldwide and so has come to be an important medium for the giant corporations to SMEs to advertise their wares.

The first chapter of this thesis gives a history of the origin of the Internet and WWW along with an analysis of the Internet protocols including HTML, HTTP and FTP. History is static and cannot be altered but the protocols used on the internet and WWW are dynamic, everchanging and evolving, with new features being added to each implementation to better accomodate either the client-side end-user or the server-side developer. HTTP1.1, for example, included many new useful facilities such as PUT and DELETE which respectively allowed the user to add or remove resources from a Web server. HTML is also improving with each version, the latest HTML4.0 adding extra support for multimedia and disabled users.

The client-server architecture of the WWW was discussed and the interaction between the client-side browser, CGI scripts and the back-end server was analysed. Obviously the action of the WWW server was looked at in more detail since it is the main focus of the thesis. The handling of browser requests by the server, the traits of a good Web site and issues such as firewall security were examined.
This area of internet security is itself a hugely important field and much research is currently taking place in an effort to make the internet and WWW more secure, essential if commerce is to prevail on the Web. Much of this research focuses on digital signature algorithms and digital certificates. The SET (Secure Electronic Transaction) protocol, endorsed by Visa and MasterCard is one of the most sophisticated uses of digital signatures to date and these companies intend to make it a standard part of e-commerce [Wayner 97]. It appears likely, therefore, that within the next few years the WWW will become a safer place to navigate and carry out transactions of a financial nature.

The most important problems associated with WWW servers were examined and of these server response time is the most pressing issue, since it has a direct effect on the oft impatient end-user. The most common contributors to poor server response times are the slow connection speed of the server and the slow speed of the user’s connection to the internet. Surveys indicate that the majority of end-users are still using very slow 14.4kbps modems to connect to the internet and will continue to do so until the prices associated with higher-speed ISDN connections drop. Currently there is much speculation in the computing world about a possible revolution in internet data transfer rates. This all stems from a device called a cable modem, which has the theoretical ability to transfer data at speeds of 30Mbps. In practice broadband speeds of 1 to 5Mbps are more realistic for the device but what is important here is the fact that is will supposedly be available for less than $40 per month, therefore within the price range of most internet users [Halfhill 96]. The device is still at testing phase and may take some time before the revolution begins.
so this author believes that slow server response time will continue to be a major issue for at least a number of years to come and may still remain an important problem since the amount of multimedia content on the Web is increasing at an uncontrollable rate, demanding valuable internet bandwidth.

6.2 WWW Software Technologies

A number of the most important Web technologies have been discussed throughout the thesis. Java, although a relatively infantile programming language, has become indispensable to the development of the WWW due mainly to its platform independence and its Java applets. Java's platform independence have made a significant difference to software developers who are now in a position where they can dedicate more time to the actual functionality of their application rather than having to worry about which platform it will execute on - perfect for the WWW where a multitude of different operating systems may connect to the same Java application. Java applets have also made an impact on the Web and have added an element of dynamism and interactivity to Web pages. In addition, Java applets offload some of the servers work as they execute on the client which downloads them. Sun are improving the Java API with each successive version of the Java Developers Kit (JDK) but it still remains to be seen whether Sun Microsystems will achieve official ISO standardisation of their Java technologies. This author believes that Sun obtaining the 'Java standard' is highly probable and may pave the way for further integration of Java with other software vendors components.
The Common Object Request Broker Architecture (CORBA) was also examined. This specification makes it possible to create generic IDL interfaces which can be used by client-server applications written in any programming language for which an IDL compiler exists. In other words, the operating system specifics are not an issue and applications can invoke methods associated with CORBA objects through an ORB in order to use their 'services'. This has huge implications for distributed computing and, of course, for the WWW which is a globally distributed network in its own right.

Much research is currently taking place in an effort to integrate CORBA with the WWW. One interesting area being investigated is an attempt to move away from the traditional Web form cum CGI script architecture in favour of a client-server 'Object Web' involving a Java plus CORBA combination. This structure has an important advantage over the traditional mechanism as it completely avoids the CGI bottleneck, allowing clients to directly invoke methods on the server via CORBA. In addition CORBA scales well and can provide load balancing for incoming client requests whereas CGI may have to respond to thousands of incoming requests with no way to distribute the load [Orfali 97]. This author firmly believes that a full-blown object Web is certainly a strong possibility for the future of the WWW but in the interim most client-server applications will remain CGI based.

The Java JDBC client-server application developed in chapter two brought together a number of the important Web technologies discussed in order to provide integration with a legacy relational database. This application was important in the sense that it showed
how the various technologies linked up but also the fact that technologies having open standards such as Java, HTML and CORBA can be utilised to integrate with legacy systems. Therefore old systems can easily be replaced with newer, more open systems, which certainly augers well for the masses of outdated applications in existence worldwide, that may need to go online eventually.

6.3 Multimedia for the WWW

The second chapter of the thesis discussed the concept of multimedia and a number of the WWW specific multimedia formats. Still images remain the primary form of multimedia on the Web but audio and video are now making a major impact also. Compression algorithms have contributed largely to this trend but the bandwidth issue discussed earlier will remain a problem for online multimedia for some time to come. The fact that large amounts of data are associated with even the smallest of movie clips means that the transfer of such information will always be a problem, especially with a slow internet connection. The point that needs to be stressed here is that the transfer problem generally lies at the client side rather than the server side and this is something WWW ‘surfers’ often tend to forget (or do not wish to admit!). The proposed OMG CORBA standard for multimedia was also analysed and if it is accepted as part of the standard it will certainly pave the way for distributed WWW multimedia applications such as Video on Demand and online video broadcasts over a multitude of operating systems (and not just Unix as has been the case with Mbone). Much research is ongoing in the area of distributed
multimedia systems, and in particular, multimedia information systems (MMIS). One such system, called HyTime provides a means for defining the representation of ‘hyperdocuments’ as character files which can be interchanged between and processed on any platform. In addition, work is taking place to allow HTML to be processed within the HyTime system in a move to prevent its obsolescence. The Distributed Multimedia Systems Lab at the University of Massachusetts continue the research in this area. In a similar vein, work is being carried out on the creation of a CORBA-based management framework for distributed multimedia services and applications. This research has involved the creation of a set of management services (defined using CORBA IDL) to monitor and control distributed multimedia applications and their supporting services and allows for the easy development of management applications [Kong 97].

6.4 Multimedia Storage on the WWW

Chapter three looked at the different modes used to store multimedia information for use on the WWW. File system storage, RDBMS, Object-relational and Object databases were investigated and compared as to their suitability in this respect. In the end, the choice of which multimedia storage mechanism to employ is completely dependent on the application in question. For simple Web pages where a non-excessive amount of multimedia data will be used for aesthetic purposes, the file system is the best approach since access to the information will be relatively fast due to the intervention of the operating system of the host machine.
Relational databases pose some problems when an attempt is made to store multimedia information within them. This information is of a complex type and relational databases were actually only invented to store 'simple' types like numbers and strings. Many DBMS vendors have a 'band-aid' solution to this problem in that they have introduced the so called Binary Large Object (BLOB) data type for storing multimedia information. Essentially, the multimedia data is stored as a digitised stream and can be reconstructed when required. This will work fine up to a point but has two significant problems. Only a small amount of data can be stored via BLOBs and the image itself cannot be queried, which is a serious flaw, where many applications are concerned. On the WWW it appears that RDBMSs will still have a significant presence, since many applications will never need to store multimedia data. Where multimedia storage, retrieval and querying is required, however, another mechanism must be used.

Object-relational databases are a relative newcomer to the database marketplace and are attempt by many of the RDBMS vendors to allow multimedia storage in relational databases. The technique used here is to take a multimedia object and literally map it to a relational database. This is a relatively time-consuming process because if an application needs to read a piece of multimedia data, the cells in the RDBMS which relate to the object must first be bundled together and then put in a particular order so that you get the final picture or sound clip. It is certainly a better solution to the RDBMS problem because the multimedia data may be queried in this case. This author believes that object-relational databases will soon become very important, despite their inefficiencies, simply because many people will wish to stay with a relational-type solution and the technology
appears to have the backing of the major RDBMS vendors. Only time and industrial performance will tell whether or not it will become the predominant mode of multimedia storage on the Web for large-scale applications.

The Object-oriented database technology has been in existence for some time and, from a multimedia perspective, its huge advantage lies in the fact that multimedia objects can be stored as complete objects so that no mapping needs to take place at all. This means that access times for object databases will typically surpass those for object-relational. In addition, it is possible to query the data. Of the techniques discussed it is certainly the best way to store multimedia information and many on-line Web applications have used this technique successfully to date. It has been proven industrially therefore. However, the odds may be stacked against it in terms of Web usage because of the parallel development of Object-relational technology, which has gained much support since its inception and which is a much more familiar alternative to people who already have RDBMS experience. If the object-relational revolution fails to transpire, then it would seem that object-oriented databases will become the predominant multimedia storage mechanism for the Web. This author believes that it could be beneficial in many ways since object databases are the only true multimedia object storage solution for pure large-scale object-oriented applications.
6.5 Development of a WWW Server Application Utilising OODBMS Technology

Drawing from the conclusions of the previous chapter, the fourth chapter of the thesis set out to investigate and utilise a specific OODBMS. The tool to be used needed to facilitate easy access of multimedia data as well as having the ability to create Web applications easily and quickly. Of the products reviewed, an OODBMS called Jasmine was eventually chosen for the application. Apart from having the normal OODBMS features, Jasmine has an easy to use graphical tool called JADE which allows for easy multimedia storage and retrieval from the database as well as an ODQL interpreter which facilitates query testing. In addition it is possible with JADE to create a multimedia database application as a regular client-server application (a distribution executable) and then to publish it as a Web-enabled application. This is an excellent feature of the OODBMS as it means that once the application is written for stand-alone purposes it can be distributed globally on the Web in a matter of minutes. No other OODBMS investigated by this author provides this feature yet. In addition, Jasmine is partnered with a number of organisations including Excalibur, which provide Jasmine integration with their product called RetrievalWare to query multimedia content within documents. This sort of support means that Jasmine can be used by diverse WWW applications from medical applications to e-commerce to plain image or sound archiving.
The actual application under development here is an educational system where the end-user can log into the system (provided their username/password combination exists in the Jasmine database). If login proves successful the user will then enter the menu 'scene' which allows them to branch off to several different paths depending on their interests. The application illustrates the power of the OODBMS in the way still images, movie clips and audio can all be stored and retrieved just as easily as the traditional data types. The ODQL interpreter is also illustrated in the application and the ability of Jasmine to integrate with standards such as ActiveX is shown. It is even possible to browse the Web from the application! The application illustrates an example of an on-line catalog of images (e-commerce) where the user can scroll through the catalog and view the details associated with each product displayed.

Once the stand-alone distribution executable had been created, the next task was to publish the application on the Web. This was relatively straight-forward in the JADE environment - the only difficulty was in modifying some operating system configuration details. In addition a plug-in for Netscape and Internet Explorer needed to be installed on the local machine and a remote terminal in order to view the Web application. The Jasmine HTTP server needed to be started in addition to the WWW server on the local machine in order to access the Web 'scenes'.

When the application was published it was tested both locally and remotely in terms of performance factors from a number of angles. These included cold and warm access time measurements, the effect of large versus small query results, the effect of the network and
the effect of network load on the performance. All of the results observed complied with expectations.

6.6 Final Conclusions

Global use of the internet has soared due to the arrival of the WWW. The attraction of online multimedia has lured even non-computing people to use the Web for a variety of purposes. This number will rise in the years to come, especially with the increase in development of e-commerce applications for the Web which will rise as authentication and security issues are sorted out.

In terms of Web server development the majority of applications are likely to remain in the standard Web-form plus CGI script architecture for some years to come, but the 'Object Web', primarily through the Java and CORBA technologies is set to alter this trend. The bandwidth issue will continue to be a problem on the WWW, especially where multimedia data is involved, and until technologies such as cable modems come into everyday use at the client-side, this will continue to be an issue. In addition, with the increase in the number of applications requiring multimedia storage and retrieval, there will be a major shift from relational to object-type DBMSs for Web applications. The only doubt is whether this trend will be towards object-relational or pure object-oriented DBMSs.
Appendix A - Noticeboard Application Code

APPENDIX A

NOTICEBOARD APPLICATION CODE
A1 - Noticeboard HTML Client

```html
<HTML>
<TITLE>Tralee RTC Staff NoticeBoard</TITLE>
<HEAD>
<BODY TEXT="#000000" BGCOLOR="#ffffff" LINK="#0F14D2" VLINK="#218D15" ALINK="#1A41A2">
 <A HREF="http://ns.rtc-tralee.ie"><IMG SRC="http://brandon.rtc-tralee.ie/gifs/newcrest.jpg" BORDER=0 HEIGHT=70 WIDTH=50 ALIGN=LEFT></A>
 <IMG SRC="/home/jbros/GIFS/earthani.gif" BORDER=0 HEIGHT=60 WIDTH=60 ALIGN=right>
 <CENTER><P>
 <HR><FONT SIZE=-1>T</FONT><FONT SIZE=-2>RTC</FONT><FONT SIZE=-1>STAFE</FONT><FONT SIZE=-2>NOTICE</FONT><FONT SIZE=-1>BOARD</FONT></CENTER>
 <CENTER><H2>WELCOME TO THE ITT STAFF NOTICEBOARD</H2>
 <IMG SRC="/home/jbros/GIFS/clip.gif" BORDER=0 HEIGHT=70 WIDTH=50 ALIGN=centrer>
 <HR></CENTER>
 <center><P>
 <TABLE BORDER CELLSPACING=1 CELLPADDING=1><TR><TD><FORM ACTION="http://skelig.rtc-tralee.ie/cgi-bin/notice.cgi" METHOD="POST">
 <i>Username</i></td><td><INPUT NAME="name" TYPE=Text SIZE=8 MAXLENGTH=8></td></TR>
 <TR><td><i>Password</i></td><td><INPUT NAME=Password TYPE=Password SIZE=8 MAXLENGTH=8></td></TR>
 <BR>
 <input type=hidden name=second value="N"></table>
 <BR><table BORDER CELLSPACING=1 CELLPADDING=1><TR><td colspan=3 align=center><INPUT NAME=Login TYPE=Submit VALUE="STAFF LOGIN"></td></TR></table>
 <HR></center>
 <BR>
 <i>If you are experiencing difficulties logging on click on</i> <a href="help.html">HELP</a>
 <HR>
 <P><center><HR></center></P>
 This page has been accessed <img src="/home/jbros/Count.cgi?df-count.dat&frgb=F08080&chcolor=T" align=absmiddle > times since 3rd June 1997.</center>
 <BR>
 <center><EM>If you are a new staff member and wish to have an account on this service, or if you have any comments on this service please contact</EM></center>
 <center><BR>
 <a href="mailto:notices@staffmail.rtc-tralee.ie">The Staff NoticeBoard Service</a> at <a href="http://ppc.rtc-tralee.ie">Tralee Regional Technical College</a>.</center>
 <EM><BR></EM><BR><center><HR></center></P>
 Comments and feedback welcomed by the <a href="team.html">NoticeBoard team</a>.</center>
</BODY>
</HTML>
```
# This cgi-bin script acts as an interface between the HTML form (Noticeboard.html) and the java application(Noticeboardapp.java). Unlike in C or C++ where the getenv() function may be used, this script is essential here since there is no other method by which the environmental variables may be passed to the java program. It does this below using the -D option after the java interpreter command.

The following line must appear on all CGI scripts implemented with the Unix shell

```
#!/bin/sh
```

The CLASSPATH environment variable contains some extra entries needed to successfully use the DataBroker5.0 server and OPENjdbc client.

CLASSPATH=/opt/DataBroker50/client/openjdbc ow10.zip:/research1/OrbixWeb2.0.1/classes:/research/jdbc/JAVAFILES:/research1/apache/cgi-bin:
export CLASSPATH

The JAVA variable just contains the location of the Java interpreter which will run the previously compiled bytecode called Noticeboardapp.class

JAVA=":/research1/jws/JDK1.1.1/jdk1.1.1/bin/java"

```
$JAVA \
```

The following set of variables are used by the GET and POST methods

- `cgi.content_type=$CONTENT_TYPE`
  - the MIME type of the data sent with the request
    - It has the datatype application/x-www-form-urlencoded here indicating data coming from a form

- `cgi.content_length=$CONTENT_LENGTH`
  - the number of bytes of information to be processed

- `cgi.request_method=$REQUEST_METHOD`
  - Either GET, HEAD or POST (POST here)

- `cgi.query_string=$QUERY_STRING`
  - The information following the ? in the URL

The following set of variables contain server information

- `cgi.server_name=$SERVER_NAME`
  - Host name or IP address of the server (skellig here)

- `cgi.server_port=$SERVER_PORT`
  - Port number to which the request was sent (80 here since it is a HTTP request)
Appendix A - Noticeboard Application Code

The following set of variables contain executable and physical path information:

-Dcgi.script_name=$SCRIPT_NAME \   #The virtual path to the script being executed
-Dcgi.path_info=$PATH_INFO \   #Extra path information passed to the script on the
#URL command line
-Dcgi.classpath=$CLASSPATH \   #All CLASSPATH information defined previously
-Dcgi.it_config_path=$IT_CONFIG_PATH \   #PATH information specific to the location of Iona
#Technologies OrbixWeb classes
NoticeboardApp   #the Java application being interpreted

A3 - JAVA APPLICATION

The following Java application has a number of tasks to perform as follows:

(a) Add the I-Kinetics OPENjdbc client driver into its jdbc drivers list for use later
(b) Attempt to make a connection to a relational Oracle database on a remote machine (skelig)
(c) Parse the form data sent with the request into a hashtable for later use
(d) When a connection has been made to Oracle, query the database and place the corresponding results
   into a ResultSet object.
(e) Compare the hashtable information i.e. the username and password the user passed in with the
   ResultSet information. If the username/password combination are present on the ResultSet, issue a
   message on the Client machine’s browser welcoming him to the ITT Staff Noticeboard, otherwise
   issue a warning message as a response.
(f) Carry out tests along the way and error and exception checking for all the code involved.

Java packages and classes which must be included:

import java.net.URL;
import java.sql.*;
import java.io.*;
import java.util.*

-A(iv)-
/the top-level application class
class NoticeboardApp {

//main() program

/*************************************************************/
public static void main (String args[]) {
/*************************************************************/

// url object will hold the URL for connection to the Oracle database required
String url="jdbc:iiop:i-kinetics:ds=oracle;dss^skelig_TRTC”;

// query object will hold the SQL query to be used on the database once connection has been obtained –
//the table called staffaccounts is being used to hold all valid username/password combinations
String query="SELECT * FROM staffaccounts”;

// Adding the OPENjdbc driver to the jdbc drivers list
Properties props=System.getProperties();
String drivers= new String("com.ikinetics.cw.sql.driver");
String olddriv=props.getProperty("jdbc.drivers”);
if (olddriv!=null)
    drivers.concat(“:"+olddriv);
props.put("jdbc.drivers",drivers);
props.getProperty(“jdbc.drivers”);

//Test information to check if everything is in order at this early stage
//It checks to ensure the OPENjdbc driver has been successfully added and that the CGI script has sent
//the environment variables

   //System.out.println(props.getProperty("jdbc.drivers”));
   //System.out.println(props.getProperty("QUERY_STRING”));
   //System.out.println(props.getProperty("SCRIPT_NAME”));
   //System.out.println(props.getProperty("SERVER_NAME”));

try {

//This sets the logging/tracing printstream to be used by the drivermanager and all drivers to the screen
//It outputs all connection related information to the browser and so is only used when testing or
//modifying the application

   //DriverManager.setLogStream(System.out);

//The next code deals with the form data passed in from the user’s browser

//Print the required CGI header.

-A(v) -
System.out.println(cgi_lib2.Header());

// Parse the form data into a Hashtable.
Hashtable form_data = cgi_lib2.ReadParse(System.in);

// The next code just tests that the user's name/password combination has been passed in successfully
// It simply echoes the information to the client browser by generating the necessary HTML formatting

// (String)form_data.get("name");
// System.out.println(cgi_lib2.HtmlTop("Hello There " + name + "!"));
// System.out.println("<h1 align=center>Hello There " + name + "!</h1>");
// System.out.println("The test shows that your name is" +
// (String)form_data.get("name");
// System.out.println("and your password is" + (String)form_data.get("Password"));

// Attempt to connect to the Oracle database using the url object previously set up along with a username
// and password combination for getting access to Oracle

Connection con = DriverManager.getConnection(url, "jbros", "rtc");

// Check for any warnings from the connection - if none are generated a successful connection has been
// made

checkForWarning(con.getWarnings());

// The next code just verifies that all name=value pairs sent from the from are correct

// System.out.println("Here are the name/value pairs from the form: ");

// Print the name/value pairs sent from the browser.

// System.out.println(cgi_lib2.Variables(form_data));

// The following checks that all the necessary environment variables have been passed from the CGI script

// System.out.println("Here are the CGI environment variables/value pairs" +
// "passed in from the UNIX script: ");

// Print the environment variables sent in from the Unix script.

// System.out.println(cgi_lib2.Environment());
Appendix A - Noticeboard Application Code

The next code allows us to get some information about the connection via the DBMetaData object. It is for testing purposes only to ensure a correct connection has been achieved:

```java
DatabaseMetaData dma = con.getMetaData();
// System.out.println("Connected to " + dma.getURL());
// System.out.println("Driver : " + dma.getDriverName());
```

Next, a statement object is created with which SQL statements may be submitted to the driver:

```java
Statement stmt = con.createStatement();
```

Submit the query previously set up and place all corresponding results in a ResultSet object:

```java
ResultSet rs = stmt.executeQuery(query);
```

The following method checks to ensure that all required information has been included in the ResultSet and checks that the username/password supplied in the form is present in the ResultSet and based on the outcome generate appropriate messages:

```java
dispResultSet(rs, form_data);
```

Now all business with the ResultSet object is finished, close it:

```java
rs.close();
```

Similarly for the Statement object:

```java
stmt.close();
```

And the Connection object:

```java
con.close();
```

Now the try clause has ended, it must have a corresponding catch clause to catch any exceptions or errors:

```java
// Catch any connection related exceptions or errors - print details to the screen/browser
catch(SQLException ex) {
    System.out.println("***SQLException caught***
    while(ex!=null) {
        System.out.println("SQLState: " + ex.getSQLState());
        System.out.println("Vendor specific error code: " + ex.getErrorCode());
    }
}
```

-A(vii) -
ex=ex.getNextException();
    System.out.println("\n");
}

//Catch any exceptions/errors related to the Java language itself

catch(java.lang.Exception ex) {
    ex.printStackTrace();
}

//Check for any warning messages generated on trying to connect to the Oracle database

private static boolean checkForWarning(SQLWarning warn) throws SQLException {
    boolean rc=false;
    //if a n SQLWarning object was given, display the warning messages. Note that there could be multiple
    //warnings chained together
    if (warn!=null){
        System.out.println("***Warning***\n");
        rc=true;
        while(warn!=null) {
            System.out.println("SQLState: "+warn.getSQLState());
            System.out.println("\n");
            warn=warn.getNextWarning();
        }//end while
    }//end if
    return rc;
} //end checkForWarning()

// The following method displays all columns and rows in the table being accessed for the given QUERY.
//It also checks if the user supplied login/password combination is present in the table. From there it
//generates appropriate output messages via HTML formatting.

private static void dispResultSet(ResultSet rs,Hashtable form data) throws SQLException {
    int i; //loop counter
    //Get the ResultSetMetaData object (used for column headings)
    ResultSetMetaData rsmd=rs.getMetaData();
    //Get the number of columns in the result set
    int numCols=rsmd.getColumnCount();
    //The next code deals with the form data passed in from the browser
boolean incorrect=true;
boolean more=rs.next();

while (more) {

//Loop thru each column, getting column data and displaying (for test purposes only)

/* for (i=1;i<=numCols;i++) {
   if(i>1)
      System.out.print",";
   System.out.print(rs.getString(i));} */
System.out.println(" ");

//If the username/password combination corresponds to information in the ResultSet object welcome
//the user to the ITT Staff Noticeboard

   if (((rs.getString(2).compareTo((String)form_data.get("name")))==0)
      &&((rs.getString(3).compareTo((String)form_data.get("Password")))==0))
   {
      System.out.println("<b>Welcome to the staff noticeboard!</b> ");
      more=false;incorrect=false;}
   else{
      more=rs.next();
      incorrect=true;}//end of if-else
//end of while loop

   if ((more==false)&(incorrect=true))
      System.out.println("<b>Warning: Either your username or password (or both)
      are incorrect!</b> ");

//Generate the HTML necessary to close a HTML document cleanly

   System.out.println(cgi_lib2.HtmlBot());

}//end dispResultSet

}//end main()
class cgi_lib2
{

//The following method is used to parse the form input information into a hashtable for easy processing

//public static Hashtable ReadParse(InputStream inStream)
/package static Hashtable ReadParse(InputStream inStream)

    
    
    
    

    
    
    
    

if (MethGet())
{
    inBuffer = System.getProperty("cgi.query_string");
}
else //if the data is sent with a method of POST as in this case
{
    DataInput d = new DataInputStream(inStream);
    String line;
    try
    {
        while((line = d.readLine()) != null)
        {
            inBuffer = inBuffer + line;
        }
    }
    catch (IOException ignored) { }
}

//Split the name/value pairs at the ampersand (&)

StringTokenizer pair_tokenizer = new StringTokenizer(inBuffer,"&");

while (pair_tokenizer.hasMoreTokens())
{
    String pair = urlDecode(pair_tokenizer.nextToken());

    //Split into key and value

    StringTokenizer keyval_tokenizer = new StringTokenizer(pair,"=");
    String key = new String();
    String value = new String();
    if (keyval_tokenizer.hasMoreTokens())
        key = keyval_tokenizer.nextToken();
    else ; // ERROR - shouldn't ever occur
    if (keyval_tokenizer.hasMoreTokens())
        value = keyval_tokenizer.nextToken();
    else ; // ERROR - shouldn't ever occur
    //
    // Add key and associated value into the form_data Hashtable
    //

-A(x)-
form_data.put(key,value);
}
return form_data;

//The following method is used directly by the ReadParse method to URL decode a string

public static String urlDecode(String in)
{
    StringBuffer out = new StringBuffer(in.length());
    int i = 0;
    int j = 0;
    while (i < in.length())
    {
        char ch = in.charAt(i);
        i++;
        if (ch == '+') ch = ' ';
        else if (ch == '%')
        {
            ch = (char)Integer.parseInt(in.substring(i,i+2), 16);
            i+=2;
        }
        out.append(ch);
        j++;
    }
    return new String(out);
}

//The following method simply generates a standard HTML header

public static String Header()
{
    return "Content-type: text/html\n\n";
}

//The following method just generates the top of a HTML page - head and title information specifically

public static String HtmlTop(String Title)
{
    String Top = new String();
    Top = "<html>
    <head>
    <title>
    Top+= Title;
    Top+= "\n";
    return Top;
}
Appendix A - Noticeboard Application Code

```java
Top+= "</title>\n;  
Top+= "</head>\n;  
Top+= "</body>\n;  
return Top;  
}

//The following code generates the bottom of a standard HTML page

/**  
 * public static String HtmlBot()  
 */  
{/  
    return "</body>\n</html>\n";  
}

//Method which determines if the method used to send the request was GET

/**  
 * public static boolean MethGet()  
 */  
{/  
    String RequestMethod = System.getProperty("cgi.request_method");  
    boolean returnVal = false;  
    if (RequestMethod != null)  
    {  
        if (RequestMethod.equals("GET") ||  
            RequestMethod.equals("get"))  
        {  
            returnVal=true;  
        }  
    }  
    return returnVal;  
}

// Method which determines if the method used to send the request was POST

/**  
 * public static boolean MethPost()  
 */  
{/  
    String RequestMethod = System.getProperty("cgi.request_method");  
    boolean returnVal = false;  
    if (RequestMethod != null)  
    {  
        if (RequestMethod.equals("POST") ||  
            RequestMethod.equals("post"))  
        {  
            returnVal=true;  
        }  
    }
```
public static String Environment()
{
    String returnString;

    returnString = "<dl compact>
    
    <dt><b>CONTENT_TYPE</b> <dd>:<i>" + System.getProperty("cgi.content_type") + "</i>:<br>
    <dt><b>CONTENT_LENGTH</b> <dd>:<i>" + System.getProperty("cgi.content_length") + "</i>:<br>
    <dt><b>REQUEST_METHOD</b> <dd>:<i>" + System.getProperty("cgi.request_method") + "</i>:<br>
    <dt><b>QUERY_STRING</b> <dd>:<i>" + System.getProperty("cgi.query_string") + "</i>:<br>
    <dt><b>SERVER_NAME</b> <dd>:<i>" + System.getProperty("cgi.server_name") + "</i>:<br>
    <dt><b>SERVER_PORT</b> <dd>:<i>" + System.getProperty("cgi.server_port") + "</i>:<br>
    <dt><b>SCRIPT_NAME</b> <dd>:<i>" + System.getProperty("cgi.script_name") + "</i>:<br>
    <dt><b>PATH_INFO</b> <dd>:<i>" + System.getProperty("cgi.path_info") + "</i>:<br>
    <dt><b>IT_CONFIG_PATH</b> <dd>:<i>" + System.getProperty("cgi.it_config_path") + "</i>:<br>
    <dt><b>CLASSPATH</b> <dd>:<i>" + System.getProperty("cgi.classpath") + "</i>:<br>
    
    </dl>
    
    return returnString;
}
public static String Variables(Hashtable form_data) {
    String returnString;

    returnString = "<dl compact>
    
    for (Enumeration e = form_data.keys(); e.hasMoreElements(); ) {
        String key = (String)e.nextElement();
        String value = (String)form_data.get(key);
        returnString += "<dt><b>" + key + "</b> <dd><i>" + value + "</i>\n"
    }

    returnString += "</dl>\n"

    return returnString;
}

} end of Variables()

} end of cgi_lib2 class
Appendix A - Noticeboard Application Code

A4 - ORACLE SQL CODE

/*The following SQL command creates the table called staffaccounts with the given column headings and
data types where VARCHAR2 is a string of varying maximum lengths (enclosed in brackets) and
NUMBER is an integer value of a certain maximum value (999 in the case of a 3 digit number here)*/

SQL> create table staffaccounts (REALNAME VARCHAR2 (25), ACCNAME VARCHAR2(8), PASSWORD VARCHAR2(8), SCHOOL VARCHAR2(15), SUBJECT VARCHAR2 (15) STAFFNUMBER NUMBER (3));

/*The following command now inserts a row of data values into the table, doing so only if the data values
comply with the column data types*/

SQL> insert into staffaccounts values('John Brosnan', 'jbros', 'wTcyz', 'Science', 'Computing', 123);

The complete structure and contents of the Oracle table is as follows:

<table>
<thead>
<tr>
<th>REALNAME</th>
<th>ACCNAME</th>
<th>PASSWORD</th>
<th>SCHOOL</th>
<th>SUBJECT</th>
<th>STAFF#</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Brosnan</td>
<td>jbros</td>
<td>wTcyz</td>
<td>Science</td>
<td>Computing</td>
<td>123</td>
</tr>
<tr>
<td>Deirdre Lillis</td>
<td>lillis</td>
<td>abcd</td>
<td>Science</td>
<td>Computing</td>
<td>234</td>
</tr>
</tbody>
</table>
APPENDIX B

OBJECT DATABASE APPLICATION CODE
package COM.odi.demo.people;

// Import the COM.odi package, which contains the ObjectStore Java API:
import COM.odi.*;
public
class Person {

// Fields in the Person class:
String name;
int age;
Person children[];

// Main:

public static void main(String argv[]) {
    String dbName = argv[0];

    // The following line initializes the ObjectStore Java software.
    ObjectStore.initialize(null, null);
    try {
        Database.open(dbName, ObjectStore.OPEN_UPDATE).destroy();
    } catch (DatabaseNotFoundException e) {
        Database db = createDatabase(dbName);
        readDatabase(db);
    }
}

static Database createDatabase(String dbName) {
    // Call the Database.create() method to create and open the
    // database that is specified on the command line
    // when the application is invoked:
    Database db;
    try {
        db = Database.open(dbName, ObjectStore.OPEN_UPDATE);
        db.destroy();
    } catch (DatabaseNotFoundException e) {
        db = Database.create(dbName, ObjectStore.ALL_READ | ObjectStore.ALL_WRITE);
    }

    // Start an update transaction:
    Transaction tr = Transaction.begin(ObjectStore.UPDATE);

    // Create instances of Person:
Person sophie = new Person("Sophie", 5, null);
Person joseph = new Person("Joseph", 1, null);
Person children[] = { sophie, joseph };

Person tim = new Person("Tim", 35, children);

// Create a database root and associate it with
// tim, which is a persistent-capable object.
// ObjectStore Java uses a database root as an entry
// point into a database.

db.createRoot("Tim", tim);

// End the transaction. This stores the three person objects,
// along with the String objects representing their names, and
// the array of children, into the database.

tr.commit();
return db;

static void readDatabase(Database db) {

// Start a read-only transaction:

Transaction tr = Transaction.begin(ObjectStore.READONLY);

// Use the "Tim" database root to access objects in the database.
// Because tim references sophie and joseph, obtaining the "Tim"
// database root allows the program to also reach sophie and joseph.
// In each transaction, an application must obtain a database root
// and use it to navigate to persistent objects.

Person tim = (Person)db.getRoot("Tim");
Person children[] = tim.getChildren();
System.out.print("Tim is "+tim.getAge()+" and has "+children.length+" children
named:"+"\n"); for (int i=0; i<children.length; i++) {
String name = children[i].getName();
System.out.print(name + " ");
System.out.println("\n");

// End the read-only transaction. This ends the
// accessibility of the persistent objects and abandons
// the transient objects.

tr.commit();
}

// Constructor:
public Person(String name, int age, Person children[]) {
this.name = name; this.age = age; this.children = children;
}

public String getName() { return name; }

-B(iii) -
public void setName(String name) { this.name = name; }
public int getAge() { return age; }
public void setAge(int age) { this.age = age; }
public Person[] getChildren() { return children; }
public void setChildren(Person children[]) {
    this.children = children;
}
APPENDIX C

MULTIMEDIA OBJECT DATABASE EDUCATIONAL SYSTEM SCENES
Welcome to the Institute of Technology Tralee Multimedia Object Database Application

If you wish to get a feel for the capabilities of an object database look no further!

Login
Password

If you do not have an account to access the object database contact the system administrator.

This database has been created using the Jasmine Version 1.1 Developer Edition software product from Computer Associates

Congratulations, you have successfully logged into the Jasmine multimedia object database application. I hope you enjoy your trip through the system and see what can be achieved using this object oriented database technology.

It is possible in JADE to use widgets such as buttons and static text boxes to add functionality and interactivity to the application. For example, I have used buttons below on the left to enable you to connect to an appropriate scene upon clicking the left mouse button. In addition, I have defined behaviour for the associated static text boxes on the right which will also allow you to connect to the appropriate scene but upon the action of a right mouse click rather than a left one.

1. Play some audio and video samples from the database store
2. Browse through a fashion catalogue
3. Carry out an interactive transaction on the database
4. Use the Jasmine OQL interpreter utility to query the database
5. Play a movie and update a MS Word document interactively using ActiveX controls
6. Browse the Web using the Microsoft Web Browser ActiveX control
7. Send any comments/queries you have to the system administrator

Exit System
Welcome to the audio/visual scene!
Prepare yourself for some real multimedia content

When most people think of multimedia they usually expect some combination of sounds and sights. Traditionally, database storage techniques have been rather limiting in the sense that the data stored was restricted to textual information such as strings and numbers. The Jasmine object database model overcomes these restrictions and stores images and sounds just as easily as relational database type data. I have set this scene up so that if you left-click on the ITT crest above, it will activate a .avi movie related to the static GIF image on the left. It will appear that the static image has disappeared but this is not true. It is simply hidden from view while the movie plays and will reappear once the movie is over.

You will probably notice that this movie is silent. If you want to get a taste for the audio capabilities of this object database technology, just left-click on the image of the cartoon character, Homer Simpson, on the left and hear him "sing"! Afterwards you may regret listening!

This scene shows a common use of a multimedia object database, an electronic commerce (e-commerce) application. In this instance you see a fashion catalogue system from where you can browse the list of items in the catalogue, see a large photo of any thumbnail image (by clicking on the thumbnail) and see a description of the details of the garments such as name, size, price etc. I have set the behaviour of the scene items so that while you are scrolling through the catalogue of images, you will be unable to see a large photo or garment details. Try it out for yourself now!
Transaction Processing in an Object Database

This scene illustrates one of the important features of any DBMS, the ability to process a transaction. In this scene you see a catalogue item displayed. You can change the price and associated "hot value" of the particular garment shown interactively by starting a transaction and left-clicking the "On Sale" button. This will immediately effect the price and popularity rating of the garment both on screen and in the database. Should you change your mind and decide that you want the old price back, click "Rollback Txn" to undo the effect of the changes. If you are happy with the change click on "Commit Txn" to make the changes permanent in the database. Notice however, that if you do not start a transaction and click the "On Sale" button you do not have the choice to undo the changes using a rollback mechanism.

Start Txn
Commit Txn
Rollback Txn

Item: Apple Red Windbreaker
Price: 2.63
Hot Value: 14

Using Interpreted Object Database Query Language (ODQL)

This scene gives you the opportunity to experience the Jasmine ODQL Interpreter. Although I have created this application using the Jasmine Application Development Environment (JADE) and the use of the Jasmine Class Browser to define classes, their properties, methods, queries and objects, there are situations where this tool is unavailable and you need to resort to the "traditional" interpreted environment. This is "back to basics" command-line stuff but it is possible to execute a file of ODQL commands to carry out more complex transactions. This scene gives you the chance to do both.

Note: to exit the ODQL interpreter program, type end at the ODQL prompt.

Sample: Suppose you wish to know the details of the person with an account in the database whose name is given by the string "brosnan". The following ODQL commands and statements retrieve this information from the object database and display it in the interpreter. Case-sensitivity is a must so make sure you type everything as shown, otherwise an error message(s) will be given. When you have retrieved the information click on the "check" button below to ensure it corresponds to the correct information presently in the database.

default CF CASStore;
Bag<Customer> cc;
cc = Customer from Customer where Customer.name == "brosnan";
cc.print();

Name: 
Address: 
Password: 
Phone no.: 
Shoe size: 
Neck size: 

Start Interpreter

Check

-C(iv) -
Welcome to the ActiveX Scene!

JADE Player - ActvX

Welcome to the ActiveX Scene. ActiveX controls are software objects which, if placed in a compatible container such as Microsoft VB or Word, works with object linking and embedding (OLE) to provide these controls for users to access data. OLE allows applications to share information and functionality within and between one another and the OLE client and server are part of the NT operating system. It is possible to set the properties of an ActiveX control to determine how it will react with the user at runtime. Below you will get a chance to experience the capabilities of ActiveX controls first hand.

You see what appears to be an ordinary image on the left. However, this is actually an ActiveX control that has special capabilities, in this case I created the control from a movie (.avi) file on the hard disk. I have set its behaviour so that if you left-click on the control it will activate the associated application - in this case MediaPlayer. You then have the options of editing the video interactively and temporarily updating it (you can try giving the clip in the scene a caption for balance) and, of course, running it also. Enjoy the goal!

Welcome to the Web Scene - Happy Browsing!

This MS Word document is an OLE object within the JADE application. Like the movie control, it is possible to left-click on the control and activate the MS Word application. This allows you to modify and update the text displayed in the document interactively. Note however that these changes will not be permanent (if you go to the Home scene, for example, and return to this scene, any changes you made will have disappeared). This is because the text file associated with the ActiveX control is that set at design time. The only way to permanently change things is to edit the Word document at the design stage. This means that you will be unable to change the OLE document permanently. Try it out, remembering to update the document within MS Word.

Welcome to
The Institute of Technology
Tralee

http://www.ittralee.ie
Thank you for your interest in this multimedia object database

Hi, my name is John Brosnan and I have developed this multimedia object database as part of my MSc. thesis by research at the Institute of Technology, Tralee. If you need a new account, have forgotten your account details or simply wish to comment on or query the system in general please contact me using one of the following modes:

- Phone (IT)
- Phone (home)
- E-mail
- Pigeon-hole
- Letter

Exit  Home
Bibliography & References


ANSAWeb 96  ‘Building a Web from Distributed Objects’, ANSAWeb, April 1996. Internet URL: [http://www.ansa.co.uk/ANSA/ISF/9606-workshop-tms.html]


BMRC 98  ‘Helper Applications - The Short Answer’, Berkeley Multimedia Research Center, Regents of the University of California, February 1998. Internet URL: [http://bmrc.berkeley.edu/info/helpers.html]


Codd 70 E.Codd. 'A Relational Data Model for Large Shared Data Banks'. Communications of the ACM, 13(6),June 1970.


Davies 95 J.Davies, R.Weeks & M.Revett. 'Jasper:Communicating Information Agents for WWW'. BT Laboratories, Ipswich, UK, 1995

Bibliography and References


Digital 96 ‘Know Your Net’, Intranet and Internet Computing for your Business, 1996 Digital Equipment Corporation. Produced by The Red Group, Banbury, UK

DMSL 96 ‘Distributed Multimedia Systems Laboratory Homepage’, DMSL, Department of Computer Science, University of Massachusetts Lowell, USA, April 1996. Internet URL: [http://dmsl.cs.uml.edu/]


Georgia Tech 97 ‘Graphic, Visualisation & Usability Center’s (GVU’s) 8th WWW User Survey’. October - November 1997. GVU’s WWW Surveying Team, GVU Center, College of Computing, Georgia Institute of Technology, Atlanta, GA, USA. Internet URL: [http://www.cc.gatech.edu]


Hethmon 97 P.Hethmon 'Illustrated Guide to HTTP'.
Internet URL: [http://www.manning.com/Hethmon]

Internet URL: [http://www.i-kinetics.com]


Internet URL: [http://www.igd.fhg.de/icib/it/iso/is_10918-1/read.html]

ISO 94b 'Generic Coding of Moving Pictures and Associated Audio', IS-13818, International Organisation for Standardisation, Case Postale 56, 1 rue de Varembe, Geneva, Switzerland, November 1994


Internet URL: [ftp://devo.cs.uml.edu/pub/hytime/ACM_MM_93_paper.ps.gz]

[Internet URL: http://amazon.postech.ac.kr/pubs-97.html]

Larner 96 D.Larner. 'Migrating the Web Toward Distributed Objects'. Xerox Parc, Palo Alto, CA, USA. 1996


Microsoft 96 Microsoft Corporation. 'Installing Internet Information Server', 1996. Internet URL: [http://www.microsoft.com]


Perlman 95 E.Perlman & I. Kallen . 'Common Internet File Formats', for Internet Literacy Consultants, December 1995. Internet URL: [http://www.matisse.net/files/formats.html]


Raggett 97 D. Raggett & A. LeHors. 'HyperText Markup Language'. Internet URL: [http://www.w3.org/MarkUp]

Internet URL: [ftp://devo.cs.uml.edu/pub/hytime/IASTED_95.ps.gz]

Rutledge 96 L.Rutledge. ‘Presenting HyTime Documents with HTML’, DMSL, Computer Science Dept., UML, USA.
Internet URL: [ftp://devo.cs.uml.edu/pub/hytime/EdMedia96.ps.gz]

Schulzrinne 97 H.Schulzrinne. ‘FAQ on the Multicast Backbone (MBONE)’, Department of Computer Science, University of Columbia, April 1997.
Internet URL: [http://www.cs.columbia.edu/~hgs/internet/mbone-faq.html]


Smith 96 B.Smith. ‘Brian Smith’s Home Page’, Assistant Professor at the Computer Science Department in Cornell University, USA, 1996.
Internet URL: [http://cs.cornell.edu/Info/Department/Annual96/Faculty/Smith.html]

Internet URL: [http://techweb.cmp.com/iw/s81/8iowoe.htm]


Internet URL: [http://java.sun.com/applets/resources.html]

Internet URL: [http://www.cs.cmu.edu/~tilt/cgh]

Internet URL [http://elib.cs.berkeley.edu/cypress]
Internet URL: [http://www.bergen.org/AAST/ComputerAnimation/]

Varela 95 C. Varela et al. 'Browsing Object-Oriented Databases over the Web'.
Department of Computer Science, 1304 West Springfield Ave., University of
Illinois, Urbana, USA, 1995

Internet URL:[http://www.w3.org/TR]

Internet URL: [http://www.w3.org/TR/REC-png-multi.html]

Wallis 95 R. Wallis. 'Lack of clarity in multimedia definitions can hinder development of
useful applications', Multimedia Group, City University Business School, London, July
1995.
Internet URL: [http://www.city.ac.uk/multimedia/ftnewslt.htm]


Wells 98 D. Wells. 'Ovum Evaluates: the Data Warehouse'. Application Development

Wiederhold 83 G. Wiederhold. 'Database Design'. McGraw-Hill Computer Science
Series, 1983.

Williams 96a H. Williams. 'Audio Formats', Digital Equipment Corporation, 1996
Internet URL: [http://ftp.digital.com/webmm/audio.html]

Williams 96b J. Williams. 'Using Helper Applications and Plug-Ins', Columbia
University, New York, USA, March 1996.
Internet URL: [http://www.columbia.edu/acis/software/netscape/helpers.html]