CHAPTER 6
DYNAMIC RECONSTRUCTION OF 3D VIRTUAL WORLD
USING A DATABASE SYSTEM

Progress is achieved in enhancing e-learning through virtual scenes on the Web. The capability of automatically adapting the content, structure, and/or presentation of a Web site to address the interests and preferences of each individual user is more and more considered as a key factor to increase user satisfaction. Significantly, VRML has been designed to operate in the context of the Internet facilitating the delivery of virtual reality content over the Internet. A fusion of Java, database and VRML technologies is used to design applications for the three-dimensional visualization of information stored in generic databases. The current VRML and database technologies impose several constraints, such as the External Application Interface implementation, database table’s optimization and structure. These constraints can be satisfied by including new behavior both in the client and the server.

Digital 3D models are a high-potential technique. Although creative, modeling is strongly rooted in knowledge acquisition. The ‘creation’ must resemble reality even if objective cognitive data are to be put to different use depending on the modeling technique adopted. 3D virtual models are a break with the past limits of projection and cross-section, and of the analogy between the sheet of paper and the surface of the wall, since they presents a world more in tune with the world as we perceive it. VR desktop techniques promise an accessible, highly visual, and interactive means of representing difficult-to-see data, opening up new ways of presenting research. VR models allow accommodating all of the current knowledge and thoughts about an object into a user interactive presentation.

The goal of this project is to design an architecture that allows the creation of dynamic worlds from any kind of database on the Internet where a user can easily and intuitively find the information he needs. Java Server Pages (JSP) is a tool for generating dynamic responses to just about any type of browser. Users typically interact with databases, enterprise systems, and Web servers. An active VR three-tier Web service is designed to acquire virtual scene requests from users connected to a Web browser, perform the virtual scene schemas process and provide them with the virtual scene results. A request is made through a web browser which calls a JSP
designate file to access the database. The database sends the requested data to the JSP, which formats the data and sends the response to the requesting browser thus dynamically generating and modifying the database-driven virtual scene. Application of parameterized template of scene enhances the inheritance and reusability of the virtual scene.

6.1 Introduction

A method of modeling active virtual world in database provides means of parameterization of all elements of the dynamically generated virtual scenes. The term "active" is used to describe applications that allow server-side user interaction, dynamic composition of virtual scenes, access to on-line data, continuous visualization, and implementation of persistency. The state of a virtual world has to be retrieved, manipulated and stored as it changes over time. This is called persistence and is gaining importance among VRML applications. As VRML applications become more complicated and process real-time data, the need for adequate persistence capabilities increases. VRML and object oriented databases are relatively new fields rapidly gaining importance and popularity.

The current VRML standards are sufficient for building passive VR systems—that is, systems that use VR technology to visualize predesigned 3D scenes. But passive systems are quite limited. Once a designer creates a virtual scene, the system stores its description in files in a ready-to-use form. A user can view it only in the original form. The virtual scene objects and their initial positions can be modified with the use of parameterized virtual scene models that form active database-driven VR applications.

The applications generate virtual scene instances according to the models, user queries, data retrieved from a database, user privileges, user preferences, and the application’s current state. Data retrieved from a database can affect all aspects of generated virtual scenes: content, visualization methods, and structure. Keeping data separate from the virtual scene model simplifies data management, allows automatic updates of the virtual scenes, and supports restrictions on user access to logical or spatial subsections of the virtual environment.

To avoid or reduce the undesirable effects of CGI scripting and facilitate management of dynamic interactions, the system can store appropriate supplementary information about behavior of objects in the database [232]. The behavior defines dynamic changes and interactions related to characteristics of objects such as shape,
position, color, etc. However, behavior can be extended to comprise changes and interactions in the virtual world. Thus, a variety of parameters, scripts, small WRL files, animations, etc. per object that facilitate and simplify the work of CGI script can be captured in the database. The result is a possibility of CGI scripts standardization, which consequently decreases their number and reduces their size. The world can be reconstructed afterwards on user request as only one script is sufficient to deliver the entire file.

Database access can occur either once during the virtual scene generation process or continuously through-out the scene lifetime. In the latter case, updates can flow from the database to the virtual scene or vice versa in response to user interaction or programmed dynamism of objects in the scene. Continuous database access supports persistency in virtual worlds. Parameters in the VRML scene models account for user queries or preferences during the virtual scene generation process. The preferences can belong to end users in the case of an Internet application based on VRML JSP is used for interpretation on the client side to change the model structure during the virtual scene runtime. The change can originate in the scene model logic, user interaction, or new data read from a database. The program can read data from a database and use values of model parameters as input. The model does not specify how many values to represent in the scene but instead reads all database values and represents them accordingly. If the number of database values changes, the virtual scene will reflect the change the next time it is rendered.

This chapter deals with the mechanisms and methods for storing, dynamically reconstructing, and navigating a three-dimensional virtual world using a database. A virtual world is described in a source text according to the grammar of VRML. The characteristics or properties of the world are represented in database tables. In an embodiment, nodes and fields of the world are associated with database queries. When the world is to be displayed, values in the database schema are recomposed into a source text. The database queries are executed against a database, yielding values, in real time based on the current state of the data in the database, for the nodes associated with the queries. Thus, a large virtual world are efficiently displayed and easily modified, and the size, shape or other aspects of the elements of the virtual world can change as data in the database changes.

VRML97 allows the description of dynamic worlds that can change with both the passage of time, and user interaction. Unfortunately, the current VRML usage
model prevents its full potential from being realized. VRML being a file format does not support database storage and querying of 3D objects. New presentations must be built from scratch while the objects required might already exist from previous works. A mechanism is analyzed that accelerates the composition procedure by providing the functionality for storing, querying 3D objects and thus dynamically (re)constructing VRML worlds based on the user's criteria.

This module adds dynamic modeling capabilities to virtual scene description. Parameterized virtual scene models are used to build active database-driven VR applications. The applications generate virtual scene instances according to the models, user queries, data retrieved from a database, user privileges, user preferences, and the application’s current state. VRML and HTML are used to develop the 3D objects and its environment. Java Server Pages (JSP) is employed to build the server side program which is responsible to generate the web pages with the combination of static and dynamic contents and to communicate with the database. Tomcat serves as the container for running the JSP web application. SQL 2005 Server, a DBMS, is used to build and manage the database for the project.

Data retrieved from a database can affect all aspects of generated virtual scenes: content, visualization methods, and structure. Keeping data separate from the virtual scene model simplifies data management, allows automatic updates of the virtual scenes, and supports restrictions on user access to logical or spatial subsections of the virtual environment. It also lets the developer model virtual environments that exceed one virtual scene either spatially or logically, as in a complex geographical information system. Database access can occur once during the virtual scene generation process or continuously through-out the scene lifetime. Updates can flow from the database in the latter case to the virtual scene or vice versa in response to user interaction or programmed dynamism of objects in the scene.

Continuous database access supports persistency in virtual worlds. Parameters in the VRML scene models account for user queries or preferences during the virtual scene generation process. The preferences can belong to end users, in the case of an Internet application based on VRML. When interpreted on the client side, VRML supports changes in a model’s structure during the virtual scene runtime. The change can originate in the scene model logic, user interaction, or new data read from a database. The program can read data from a database and use values of model parameters as input.
6.2 Overview of Java Server Pages Technology

To build an interactive and dynamic web site, a server side scripting or web application will be needed [233]. The server side support is responsible to generate dynamic contents of the web pages, handle the data in the forms sent by users and access database for retrieving or storing data for the web site [234]. Java Server Pages is one of the popular server side programming technologies in web development. To process Java Server Pages, a web server or a container is needed. Here, Tomcat is employed as a small stand-alone server for development of this project. Tomcat is an open source, Java based Web application container that runs servlet and JSP Web applications. Tomcat is supported and maintained under Apache-Jakarta subproject by volunteers from the open source Java community. There is a need to augment VRML with Java and JSP codes in order to permit interaction with database.

Java Server Pages (JSP) technology allows to easily create web content that has both static and dynamic components. JSP technology makes available all the dynamic capabilities of Java Servlet technology but provides a more natural approach to creating static content. JSP technology also contains an API that is used by developers of web containers. The main features of JSP technology are as follows:

- A language for developing JSP pages, which are text-based documents that describe how to process a request and construct a response
- An expression language for accessing server-side objects
- Mechanisms for defining extensions to the JSP language

JSP is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications. JSP have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. Java Server Pages (JSP) is a technology for developing web pages that support dynamic content which helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <%= and end with %>. A Java Server Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands. JSP is used to collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.

JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components,
passing control between pages and sharing information between requests, pages etc. Java Server Pages often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But JSP offer several advantages in comparison with the CGI.

- Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having a separate CGI files.
- JSP are always compiled before it's processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.
- Java Server Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP etc.
- JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

Finally, JSP is an integral part of Java EE, a complete platform for enterprise class applications. This means that JSP can play a part in the simplest applications to the most complex and demanding. Advantages of using JSP over other technologies:

- **vs. Active Server Pages (ASP):** The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.
- **vs. Pure Servlets:** It is more convenient to write (and to modify!) regular HTML than to have plenty of println statements that generate the HTML.
- **vs. Server-Side Includes (SSI):** SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.
- **vs. JavaScript:** JavaScript can generate HTML dynamically on the client but can hardly interact with the web server to perform complex tasks like database access and image processing etc.
- **vs. Static HTML:** Regular HTML, of course, cannot contain dynamic information.

Java is designed to thrive in a heterogeneous, networked environment. The Java Virtual Machine, the Java interpreter, exists on almost every architecture and operating system. The portability of Java is designed to be complete; a program
written on a Solaris workstation should work in the same way on a PC or a Macintosh without any modification. This portability extends to the graphical user interface and the different library components (such as the network classes). Java has built-in support for threads, and allows easy creation of applications that require performing multiple tasks simultaneously, such as checking the network communication at the same time as executing the other parts of the program.

A number of Web Servers that support Java Server Pages and Servlets development are available in the market. Some web servers are freely downloadable and Tomcat is one of them. Apache Tomcat is an open source software implementation of the Java Server Pages and Servlet technologies and can act as a standalone server for testing JSP and Servlets and can be integrated with the Apache Web Server. The web server needs a JSP engine, a container to process JSP pages. The JSP container is responsible for intercepting requests for JSP pages. Apache has built-in JSP container to support JSP pages development. A JSP container works with the Web server to provide the runtime environment and other services a JSP needs. It knows how to understand the special elements that are part of JSPs.

6.2.1 JSP Processing

The following steps explain how the web server creates the web page using JSP:

- The client browser sends an HTTP request to the web server.
- The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with .jsp instead of .html.
- The JSP engine loads the JSP page from the server and converts it into a servlet content. This conversion is very simple in which all template text is converted to println( ) statements and all JSP elements are converted to Java code that implements the corresponding dynamic behavior of the page.
- The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
- A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format, which the servlet engine passes to the web server inside an HTTP response.
- The web server forwards the HTTP response to the client browser in terms of static HTML content.
Finally web browser handles the dynamically generated HTML page inside the HTTP response exactly as if it were a static page.

6.3 Need for Database

Until today the figurative model on which every real world representation is brought back has always been based on three types of analysis:

a. reduction from 3D to 2D;

b. starting from few discreet points, construction of the entire continuous system through interpolation, usually linear;

c. Recomposing of the whole world scene through fixed images.

To improve this process it is necessary to find read and write methods capable of preserving the continuous and creating aggregation systems that make data as much unifiable and transferable as possible. A 3D model can be seen as a large, ordered database of spatial information, and it can be added to and altered over time. 3D models are, by their nature, highly intuitive interfaces to information, allowing the user to access an object directly via its 3D representation rather than via its name in a traditional text field. A wide range of further information can be obtained by simply clicking on a part of a 3D model.

Information stored within databases can be visualized by means of a dynamic generation of three-dimensional scenes. Today the dynamic generation of two-dimensional HTML pages is a standard functionality of all database systems. Therefore, the idea outlined in this chapter is to generate VRML scenes in a similar way. To achieve this goal, traditional Database Management Systems (DBMS) are used as a basis to provide the foundational functionality for VR information management and information visualization. By this means, base functionality for security and persistence is already available. Therefore changes of values are persistent for future use and are also available to other users [235]. But this requires that the client can directly communicate with the DBMS to access VRML and other data stored in its databases.

Object relational database systems are a good choice because they can be extended with new data types and operations in a similar way like object oriented DBMS but still provide sufficient support for handling relational data in the traditional way [236]. As most structured information is still stored in relational DBMS this choice of development platform is consequent. The described DBMS extension components can make changes to VRML scene values persistent but for
true persistence the client changes to the scene must be sending to the DBMS to initiate an update in the corresponding database. It will also have the ability to execute all sorts of SQL statements and to generate new VRML nodes out of the corresponding results.

A complicated world defined in a large VRML file is difficult to maintain and update. Modifications to a world necessitates retrieving of the entire VRML file, find the portion of the file in which changes must be made, update the file using a text editor and save the updated file as a single unit. In addition, the same process of editing, saving, and reloading must be used when the user wishes to add new elements to the world. Manipulating a VRML file in this manner excessively burdens the user and the computing system, and is slow and error-prone.

Database systems provide a powerful way to organize large amounts of information and retrieve selected information. They operate rapidly and can be applied to real-time data-entry situations in which data is constantly entered, retrieved, and updated. However, in the past, adapting databases to particular applications has been slow and cumbersome, requiring custom code. Database technology has tremendous advantages that can be combined with 3D technology: data abstraction and data independence, robustness from crash recovery and the transaction model, concurrency control and efficient access to large data sets.

In addition, database systems generally have limited report-generating capabilities especially with respect to delivering information over the Web. Most data is displayed in tabular reports, or other line-oriented and column-oriented methods. While some database systems provide graphic display capabilities, the displays are usually limited to simple graphs and charts. In most database systems, graphs and charts cannot be prepared using rapidly changing data from a database that forms a part of a real-time application. Usually, to generate a graph the user must specify a range of data in a table, and invoke a graph generating utility. While the graph is displayed, changes may occur in the data, but the changes are not reflected in the graph in real time. This limits the usefulness of the graphical display features. There is a clear need for a system, process and product that

- Provides rapid and efficient creation, modification and updating of a virtual world.
• Displays a virtual world efficiently and effectively on a remote display device that has limited local storage space.
• Allows elements or information to be inserted dynamically into a virtual world while the world is displayed.
• Permits elements of a graphical world to be modified in real time based upon a changing source of data.
• Permits such modification based upon information retrieved in real time from a table of a database management system.

Furthermore, DBMS are also considered helpful to achieve scalability of VRML scenes. Providing a DBMS with VRML data management capabilities make an optimized streaming of any sub scene possible. Also a user defined logical scaling with respect to application requirements is possible because the structure of the overall scene with respect to its composing components is accessible. In addition to enabling logical and physical scaling, the security system of the DBMS allows controllable access to objects, scene parts or whole scenes. While persistence and security are basic properties of a DBMS, it also has to enable permanent storage of VRML scenes or parts of them together with related metadata in a well structured way. In addition the DBMS has to provide functionalities for retrieving and manipulating VRML data. If this is achieved, it becomes possible to work with VRML scenes in DBMS supported applications as with any other data type. Due to the fact that most stored information is not of a three-dimensional nature, the system must also be able to generate new VRML scenes out of, e.g., operational business data represented by traditional data types.

Beside the above outlined persistence properties, an event handling system is necessary to support interaction between the VRML scene and the DBMS or the VRML scene and users. Therefore if, e.g., users or application programs change some data which should results in changes of the scene, the database should send an event to all corresponding users of that VRML scene. The VRML event handling within the scene can then react in an appropriate way by, e.g., changing the scene. Such a reaction can be to reload a part or the whole scene. Another possibility is only to reload the data to be visualized instead of a VRML scene. In this case the client must be able to generate VRML data out of the operational data that is to be visualized.
This means, that the client has to access the database directly and has to calculate the new parts of the scene from the received data result during the runtime of the scene.

6.3 Database Connectivity

Access to a database and to Java applications is possible by JDBC technology based on the ODBC. ODBC enables direct connection, making two discrete systems integrated. ODBC is the abbreviation for Open DataBase Connectivity, a standard database access method developed by Microsoft Corporation. ODBC was developed to create a single standard for database access in the Windows environment. ODBC makes it possible to access any data from any application, regardless of which database management system (DBMS) handles the data. ODBC inserts a database driver as a middle layer between an application and the DBMS. The purpose of this layer is to translate the application's data queries into commands that the DBMS understands. This is possible only when both the application and the DBMS are ODBC-compliant - that is, the application must be capable of issuing ODBC commands and the DBMS must be capable of responding to them. With ODBC, application developers can allow an application to concurrently access, view, and modify data from multiple, diverse databases.

ODBC does not translate well into the Java world. An interface is needed to bridge Java applications access to any database management system (DBMS) that supports ODBC. Java Database Connectivity (JDBC) is a standard database access method designed specifically for Java programs. Thus JDBC serves as a mechanical joint between programming code objects and project database entities. It’s a Java API that enables Java programs to execute SQL statements and allows Java programs to interact with any ODBC compliant database, or any DBMS supporting a JDBC driver. JDBC drivers fit into the following categories:

- JDBC-ODBC bridge provides JDBC access via most ODBC drivers
- A native-API partly-Java driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, DB2, or other DBMS
- A net-protocol all-Java driver translates JDBC calls into a DBMS-independent net protocol which is then translated to a DBMS protocol by a server
- A native-protocol all-Java driver converts JDBC calls into the network protocol used by DBMSs directly
The database connectivity for Java applications is shown in figure 6.1. An SQL-level API means that JDBC allows constructing SQL statements and embedding them inside Java API calls. This API provides programmers with a uniform interface to a wide range of relational databases, and provides a common base on which higher level tools and interfaces can be built. In short, JDBC translates between the world of the database and the world of the Java application. Thus, JDBC makes it possible to write a DBMS independent database application.

![Database Connectivity Diagram](image)

Figure 6.1 Database Connectivity for Java Applications

### 6.4 Integration of VRML with a Database

Existing integration of VRML with a Database do not supply functions which integrate with other programming languages. It is complicated to reconstruct a virtual object whenever it changes. VRML do not have the function of using other data formats in conjunction with VRML. Existing virtual object authoring tools are focused on creating virtual objects and do not supply functions to integrate other programming languages to itself. They cannot automatically make a connection between virtual object and database. Another mechanism is needed to insert database code into VRML code. The problem lies in the difficulty to dynamically control virtual object because the code needs to be changed on the execution of the SQL statement.

To enhance the functions of VRML, an easy connection method to the database is established without modifying the VRML code through Java VRML.
connectivity. The problem of security is solved as the SQL statements are run on the server using EAI and Client/Server model when virtual objects connect to the database. This is implemented through VRML, Java, Java Database Connectivity (JDBC) and the External Authoring Interface (EAI). Java Server Pages (JSP) is a tool for generating dynamic responses to just about any type of browser. This tool is for HTML content generation and do not fit the structure of 3D scenes. They do not provide functionality for 3D content. Moreover, their use is limited to server side in most cases. The main objective of JSP was to separate presentation and business logic. JSP allows for the direct insertion of VRML code into HTML. No compilation is required for JSP. A JSP file is translated into a wrl file the first time it is invoked by a client.

This project uses JSP to create dynamic websites particularly for database driven VRML worlds. The JSP application must establish a proper database connection, retrieve the data stored in the database, insert them in appropriate places of the VRML code and return the generated VRML in a way that is understandable by the browser. So the Web server needs to be configured with the correct MIME extension. VRML files have an extension as .wrl and are of type x-world/x-vrml. The web server used is Tomcat and it is quite likely that the Web server has already been set up with the correct MIME extension.

An extract from Tomcat’s web.xml file:

```xml
<mime-mapping>
    <extension>wrl</extension>
    <mime-type>x-world/x-vrml</mime-type>
</mime-mapping>
```

The server strips all JSP code when sending a response. So, on lines where only JSP code is present, the server simply sends blank lines back to the browser. This means that if we start the JSP file with only `<%@ page language="java" %>`, the VRML browser may not be able to correctly display the page and could report an error instead.

It is necessary to include both JSP and VRML headers on the same line and the content type must also be changed as follows:

```xml
<%@ page language="java" %>
<%response.setContentType("x-world/x-vrml");%>
VRML V2.0 utf8
A proper package should be imported (Java.sql.*) for database connection in JSP as shown.

```
<%@ page import="java.sql.*" %>

SQL 2005 Server is the database used. A JSP string used to set up the database connectivity is shown in the following code.

```java
String fullConnectionString = "jdbc:odbc:xxx";
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection connection=DriverManager.getConnection(fullConnectionString);
```

Figure 6.2 shows a standard request and response processes. The page is requested via a browser. The request calls the designated JSP, which interacts with a database. After receiving the request, the database sends the requested data to the JSP, which formats the data and sends the response to the requesting browser, the response being a VRML scene generated using the data from the database.

![Diagram](image)

**Figure 6.2 Standard request and response processes**

Stored information in databases is not of a three-dimensional nature. The system designed must be able to generate VRML scenes from the table. The DBMS has to provide functionalities for retrieving and manipulating VRML data. The data retrieved from the database must be formatted or recalculated to match a strict VRML grammar. A field in the database is needed to represent all the “concepts” present in the database. A number (ID) must be associated with each concept. The VR model is indeed dependent on the structure of the database, and choosing which field is associated to each ID helps to know what is used for the main representation in 3D.

An event handling system is necessary to support interaction between the VRML scene and the DBMS or the VRML scene and users. If users or application programs change some data it should result in changes of the scene where the
database should send an event to all corresponding users of that VRML scene. DBMS trigger mechanisms needs another new node which is implemented as a VRML prototype. It receives events from the DBMS and forwards them to the nodes of the scene. To achieve this, the event routing is done applying the standard VRML routing mechanism. Work in this chapter includes object-oriented language extensions, parameterization and database access methods.

6.5 System Architecture

![Diagram of System Architecture](image)

Figure 6.3 Architecture of Dynamic Modeling of Virtual Worlds from database

The architecture includes software on the server side that provides access to information and attributes and software on the client side that defines a user interface and provides rendering capabilities. The client side software is implemented using the Virtual Reality Modeling Language (VRML) that provides the rendering and interactive capabilities. Current real-time systems have been planned for visualization and a system must be capable of handling data supporting query processing in addition to dynamic user interaction according to three classes:
a. Orientation and Navigation,
b. Selection and Query,
c. Manipulation and Analysis.

The interrelationship of these three components is illustrated in Figure 6.3. Our objective is to provide a set of tools that automatically generates a visual representation of a database’s content according to user queries. Users should be able to ask for information from the database, which is converted to an SQL query and mapped to a 3D representation. An initial VR world is displayed as a visual representation of database content. The user can choose the level of detail to change the properties of the displayed 3-D object. The application is designed on three tier client server architecture. The first tier is a database server that contains domain data to set the context for the world. The second tier is the web server to satisfy user requests. The third tier is the client application that renders the world. The overall structure of the system comprises a Web browser with a VR plug-in on the client site and a Web server and a database system on the server site as in Figure 6.4.

![Client-server architecture for a 3D Visualization on the Web](image)

The system relies on HTML documents for the composition of queries and visualization of data other than 3D graphics. CGI scripts establish the protocol between the client and the server. They are responsible for the assembling of SQL queries, the access to the RDBMS and the creation of documents (HTML or VRML)
on the fly with respect to the result of the query. On the server side, a web server serves all client requests for static and dynamic content. The major components of the system are:

a. the central database engine
b. The VR central system.

The dynamic modeling technique enables the development of dynamic database-driven VR applications by building parameterized models (templates) of virtual scenes that constitute the application, and dynamic generation of instances of virtual scenes based on the models, data retrieved from a database, current values of model parameters, input provided by a user, and user privileges or preferences. The visualization and query architecture consists primarily of several generic functional modules coupled with an application specific GUI and objects and is a typical client/server architecture based on four key-components:

- open relational database management system (RDBMS)
- Web server
- Web browser, VR browser
- Server side web-oriented language and communication software to dynamically retrieve the contents.

Web servers allow serving content over the Internet using the Hyper Text Markup Language (HTML). The Web server accepts requests from browsers like Internet Explorer and then returns the appropriate HTML documents. Distributing the application between client and server across the Internet enables remote access to the information in ways not easily possible with conventional applications. Changes in server based information can be made immediately and universally available. The client side application includes the dynamic content that is requested by the user. Furthermore, the dynamic content has embedded JSP calls to update back to the server.

At startup, the user requests the URL for the world. In response, the server provides both static information of stored table contents and dynamic content that is assembled in together to create the requested world. To be successfully connected to a server machine, first, a Java source file is needed to initialize connection sockets and ports in the server machine. Another frame is displayed to modify the contents of the table and update it dynamically. The user has the ability to make live updates which are immediately rendered in the world. The domain data in the database encompasses
the different data components that are necessary to create a credible visualization. Indeed, the domain data is what ultimately makes a particular deployment useful. Providing a mechanism by which this data is seamlessly and automatically integrated together broadens the utility of the architecture.

The process of client-server communication can be described as follows: the client sends a request for information to the http server using the VR / HTML browsers, the Web server processes the request and returns the demanded data statically or dynamically to the client station. Depending on the request and the type of the data received, four fundamental phases can be distinguished: identification, query, visualization and navigation, manipulation. The requested information, in form of HTML / VRML document, e.g. a table with text data and a 3D model of is visualized in HTML / VR browsers at the client station. In the second phase, the user only retrieves information. Finally, an application running on a client computer should display the model in an interactive and intuitive way without omitting any information.

Designed for use over the Internet with a VR browser, the system is also intended as a visualization tool and as a study aid. The general approach followed is for the server to accept queries from client machine and then perform server side processing to satisfy the request. Each modification requires the entire world to be reloaded. For small data sets, reloading is not a significant issue; however, reloading large visualizations can add undesirable latencies. The client applications are stand-alone applications that support live updates, which is the ability to add features to the world without reload. The server processing provides the opportunity to enforce access control, selection of the content to be viewed, and generation of content dynamically. It implements support for the dynamic generation of VRML scenes out of operational data as, e.g., traditional relational data, multimedia data or VRML data itself.

Database modeling of virtual worlds is a technique that extends the dynamic content modeling with the possibility of representing all elements of the virtual world model in a database. Every time a user wants to access a virtual scene, a request is sent to the server. The server reads the JSP file, interprets it and dynamically creates “on the-fly” code of the virtual scene. The user can use the virtual scene in exactly the same way as a virtual scene retrieved from a file. The JSP file contains parameters that influence the generation of virtual scenes. They are provided by the user or
retrieved from a data repository, user privileges and up-to-date data read from one or several databases. This method separates the process of programming from the process of designing virtual scenes. These tasks are different in nature and performed by different tools.

A key challenge in creating visualizations is being able to provide timely, up to date information to clients automatically. In order to achieve this flexibility, the server needed to be able to determine what information was available on the server and then present the appropriate choices to the user. The server architecture was built upon the Apache web server. Apache provides all of the necessary capabilities to serve files on the Internet. It enables the coding of servlets, light weight Java methods, for access control and session control, generate dynamic content. The server architecture was designed to make update of current and development of alternate deployments as easy as possible. It adds significant capabilities for integrating dynamic content. When the server is updated live, interlocks can be implemented to protect against incoherencies during the update time. The server assembles the information and makes whatever conversions are necessary to include the information in the world. Some of the conversion processes are implementation dependent, requiring interface with the client application.

Once the client has rendered the world, the user explores the world. At various times, information is passed back to the server to provide a record of certain events, such as the addition of a feature. The client is written entirely in VRML and salient aspects of the architecture are described here. The purpose is to render the world and allow the user the ability to navigate through and interact with the world. In addition to managing the direct interactions with the user, the client architecture must also manage the information provided by the user and also to communicate with the server. In order to manage live updates in a VRML world, a URL containing VRML must be retrieved and then inserted into the world [237]. The insertion is simple provided it is permanent. If the ability to modify inserted content is required, a resource manager is necessary to manage the insertion of the content. The resource manager maintains tables for all addable object types, all objects that have been added, manages initialization of new objects and connects new content into the world.

Browser applications are typically limited in what they are capable of doing to protect the user from malicious or inadvertent modification of data on the client machine. Thus, in order to capture information from a user session, communication
with the server is necessary. The client application communicates with the server by requesting URLs for servlets. These servlets may direct the server to log the information or may also be a request for VRML content that can be subsequently be added to the world. The client has a plug-able engine that manages the representation of the 3D information coming from the server. The engine receives instructions from the server to display the generated world. The client must abstract the database as much as possible, as the user can ask any kind of database to be displayed in various ways. Therefore, the client architecture must be able to adapt to the server needs and possibilities. The client is just displaying forms about which it has no information, registering VRML nodes and waiting for user events.

A Structured Query Language (SQL) statement is configured to carry out the database operation on the database. The field values of VRML nodes are stored in the database. JSP receives these values as parameters and insert them in the appropriate places in the VRML code. Features include storing the field value, declaring a variable and storing the variable in association with the statement. The value of the variable is provided to the statement when the statement is executed by displaying the virtual world. The process method comprises of selecting the field value of the virtual world and defines the statement to generate a custom node associated with the field value when the statement is executed by delivering it to the client. Script nodes are necessary for interaction which needs prototypes to be defined.

Prototyping enables the author of a VRML World to collect nodes and routes and give them a standard interface; authors can also create nodes that extend the capability of the built-in VRML nodes. Prototyped or extended nodes are added to a World and used like any other node. The VRML interpreter or browser will first interpret protos on which other protos depend. A live update is the insertion of content into a world in such a way that the world does not need to be reloaded. Dynamic content is the generation of content “on the fly” meaning that before a particular request for content, no file exists having the requested content. The content is synthesized on the server from foreign data sources and user configuration data.

There are two different classes of queries, both requiring interactive specification. The first is focused on getting additional information about a selected object. Using VRML this can be done dynamically by script nodes. An object identifier is used to identify the selected object outside the VRML scene graph such as
posing a query to the database. The second kind of data query works in the opposite direction. The user searches objects which meet specific conditions. In this case, the query results in a set of objects [238].

Changes can be done directly at a DBMS level, using software that controls data access on the server, such as a Web server side script or Javascript or Java applets, and a new VRML or HTML document showing the required changes. Dynamically created VRML and HTML documents provide the Graphical User Interface (GUI) to complete queries, visualize results and explore 3D models. The External Authoring Interface (EAI), linking Java application to the 3D VRML scene is used to program Java applications which utilize an interactive user interface using VRML. EAI defines the set of functionality of the VRML browser that the external environment can access. In this way VRML files can be dynamically built and updated via the EAI and, in turn, the applet’s data can be also dynamically updated through the VRML interface.

6.6 System Implementation

The dynamic generation of HTML pages is a standard functionality of all commercial database systems. This feature has proven to be a very effective and practical approach to support the two dimensional visualization of data stored within database systems. A similar functionality can be realized to dynamically generate VRML scenes from a DBMS. Many of the limitations of static VRML scenes are overcome in this approach by exploiting the persistence, scalability and security mechanisms of DBMS. The geometry of VRML nodes is stored in the database with a unique ID. The issue has three central aspects:

- organization of data on the server,
- means to formulate queries,
- Visualization of the resulting information (basically 3D graphics, text) on the client station.

The dynamic reconstruction of 3D Virtual World using a Database System needs a web server that supports live updates. Apache Tomcat Server 7.0.8 is used to serve the purpose and JSP is used to communicate between the client and database on a server. SQL Server 2005 is the database used and the browser plug-in used to view the VR world is Cosmo Player. A flowchart depicting the processes involved in the construction of the dynamic VR world from the contents of a database. The dynamic reconstruction of 3D for a set of spheres and another set of beams is illustrated and
two database tables are used. Figure 6.5 is a flowchart depicting the process involved during the dynamic construction of VR world from database.

![Flowchart](image)

**Figure 6.5 Flowchart depicting the process involved during the dynamic construction of VR world from database**

### 6.6.1 Dynamic reconstruction of spheres

Fields in a VR world of construction are stored in a database along with an object identifier. Table used for reconstruction of spheres contains id, color and radius as fields. For a given id, the data of the sphere’s color (*diffuseColor R G B*), and its radius can be changed in a HTML frame and the changed data is immediately updated in the database which is displayed on the left top frame and the VR world is also rendered immediately in the right frame. The SQL statement used to retrieve data from the table spheres is to return all data contained in the table:

```java
ResultSet resultset = statement.executeQuery("select * from spheres");
```

Once the resultset is assembled, the code loops through all of the records and displays as many spheres as there are records in the database the values from the database are inserted into the VRML. Two strings, color and radius, are set to the values from the
database and then displayed in the VRML code. The spheres position in space (translation X Y Z) is handled by an integer (count), which keeps track of the number of records and displaces the sphere by 15 on the x-axis every iteration of the loop.

Figure 6.6 shows the snapshot of the output generated during the dynamic construction of the world from the data in the database table. The window is divided into three frames consisting of two columns. The first column is further divided into two row frames. The top frame consists of the contents of the database table that is rendered as a VR world in the second column frame. The bottom frame is a form to modify the values of the parameters in the database table. Values can be filled in and the Save button must be clicked to notify the changes made in the database table and in the display of the VR world.

Figure 6.6 Dynamic construction of VR world from database

Once a world has been displayed it can be made dynamic imported by associating SQL statements with fields in the world. Parameters are associated with an SQL statement and values are passed to the parameters through the URL that identifies a world to be rendered. The Save button should be clicked to save the changes in the table. The program updates the respective row that is identified by the id field. JSP generates the VRML code in the form of text output. Figure 6.7 shows the input form which is displayed on the right lower frame of the window with data for updating the database. This code is placed into the generated VRML world as the field value.
<% int count = 0;
while(resultset.next()) {
%>
DEF sphere<%=count%> Transform {
translation <%=count*15%> 0 0
children [
Shape {
appearance Appearance {
material Material {
diffuseColor <%=resultset.getString(2)%> }
}
geometry Sphere {
radius <%=resultset.getString(3)%>
} ] }
}<% count++;
}%>
statement.close();
connection.close();
}catch (java.lang.Exception ex){out.print(ex.toString()); }%>
The JSP code that generates the VRML code is shown above. The description of the scene is the first part of the visualization process. Yet the VRML document has to be displayed on the screen by the VR browser. The third dimension is created by extrusion of a parameter (geometric or thematic) of the 2D database. The communication between the server and the client is controlled by JSP. The VRML document is delivered initially as a script, which controls the relation between ID of objects and VRML nodes. Thus the user can identify the objects, query them and visualize results. The JSP code used to generate the update query when the values of the fields are changed is given:

```jsp
<
    String s1=request.getParameter("id");
    String s2=request.getParameter("color");
    String s3=request.getParameter("radius");
    Statement s=null;
    Connection cn=null;
    String fullConnectionString = "jdbc:odbc:xxx";
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    cn=DriverManager.getConnection(fullConnectionString);
    String str = "update spheres set color='"+s2+"', radius='"+s3+"' where id='"+s1+"'";
    s=cn.createStatement();
    s.executeUpdate(str);
%>
```

Figure 6.8 shows the VR world that is automatically rendered for every one second that retrieves the updated values from the database. Similar SQL queries to add new records and delete records can be implemented along with this program.
6.6.2 Dynamic reconstruction of beams

Similar to dynamic reconstruction of spheres, the construction of steel beams can also be reconstructed dynamically from the data stored in a database. Fields used in the table beam are id, color, length, breadth, depth, thickness and thicknessflange. The field id is mainly for accessing the records and is unique. The color and length fields are for storing the diffuseColor (R G B) vector and the length of the beam respectively. The breadth and depth of the cross section are stored in the fields breadth and depth. The thickness of the cross section web is stored in the field thickness and the thickness of the cross section flanges is stored in the field thicknessflange. Figure 6.9 shows the snapshot of the database table, the output generated during the dynamic construction of the world from the data in the database table and the frame to modify data in the table.

The client is kept independent of the database and its contents. The user interface dialog frame is used to make the query and the SQL statement is generated in the server and sent to the client which just displays them. Once the query is executed, the VRML is generated according to the results extracted from the database. A JSP file is used to bridge between the “concepts” stored in the database and the VMRL prototype, and between the attributes names and the prototype argument names thus generating three-dimensional visualization. The resulting VRML is sent to the client where it is cached and displayed by the VRML browser.
Figure 6.9 Dynamic construction of Beams from database content