PHASE III: Analysis, Design and Implementation of the OlympiChronicles DB System

OlympiChronicles
dBoss company

Craig Shapiro
Steffanie Orellana

2. ENVIRONMENT AND REQUIREMENT ANALYSIS

2.1 Purpose of Document

The purpose of this document is to provide detailed requirements and design specifications as well as to describe the implementation process and result for the enterprise project **OlympiChronicles** by the **DBOSS Company.** In this document there is a description of the ETL (Extract-Transform-Load) tool and process, a description of the information needs and activities within the project, the boundary of the design, the assumptions and limitations encountered throughout the course of the design and development phases. This document also contains a top-level information flow chart, as well as diagrams describing the logical flow of the various subsections of the enterprise, the different tasks and task forms, and the document forms that will be used, a description of the conceptual model, using an ER diagram, and the logical level, through the relational model, for the OlympicsDB. In this document there is also included the relation schema that will represent the data, a discussion about the normalization of these relations, any functional dependencies derived and a detail discussion of the implementation of the database and web-interface. This document is intended for staff members of CMSC424 who will be reviewing and approving this analysis. The deliverable from this phase is a 'Project Report' describing the specification of the solution and working demo of the implementation of the system.

2.2 Purpose of the Project

The purpose of the project is to design a reliable and efficient ETL tool (or master the use of an existing one) which will extract Summer Olympic facts from the web, transform the extracted data ("cleanse it" and/or format it), and load it into the OlympicsDB. This is the first goal of the **OlympiChronicles** project. The second goal is to provide users a web-accessible database of summer Olympic facts from 1896 to the present. The users will select various aspects about the games (queries) and **OlympiChronicles** will return a formatted table of the results of the user's query.

The purpose of this phase of the project is to implement the **OlympiChronicles** system following the requirements and design specifications of the early stages of the analysis and design process, to produce a working demo. The implementation includes

the population of the database and the creation of a web-interface for users to interact and query the database. The second purpose of this phase is to produce a detailed report of the entire development process: analysis, design and implementation.

2.3 Scope

The scope of this project involves multiple tasks. The first task involves researching and collecting web pages containing data about the history of the summer Olympic Games from the internet. The second includes extracting the relevant data, filtering it and populating and maintaining a web-accessible database for the summer Olympic Games from 1896 to the present, namely the OlympicsDB. The third involves creating a web interface from which the user will create a query to search for summer Olympics information based on year, location, countries in attendance, events, participants, medals, records broken, current records, and audio-visuals from the various years, which include images of the medals and posters for each Olympic year, and the flags and anthems for participating countries. Included in this task will be code to process and interpret the above mentioned queries and provide results to the user.

2.4 Assumptions

The assumptions for this enterprise are as follows:

- The user reads and understands English.
- The user does not have to subscribe to the system.
- The user will have internet access.
- The user has basic web browsing skills to access the web interface.
- The data will be accurate, reliable and complete.
- It is assumed that the database server is configured appropriately to handle the user demands placed on the project.
- There will be enough space on the Oracle server to store the data.

2.5 Technical and Conceptual Problems and Solutions.

2.5.1 Technical Problems and Solutions

Problem: The current limited knowledge of the designers in the required database language and the server-client programming.
Solution: Research to acquire the necessary knowledge to carry out these tasks and learn how to design and populate the SQL database and engineer it to be queried in a server-client environment.

Problem: Gathering the data

Solution: Research various websites which currently contain Olympic data.

Problem: Extracting the data.
Solution: To analyze each page containing embedded data; create an application which will pattern match and extract that data.

Problem: Transforming the data
Solution: Reformatting the data extracted to be loaded into the database.

Problem: Consistency of numeric data.
Solution: Convert all measurements to metrics, where applicable.

Problem: The Apache web server on the dc cluster account is not compatible with JSP.
Solution: To install and configure Tomcat.

Problem: Lack of knowledge creating web server scripts.
Solutions: Research and learn JSP.

Problem: Configuring RoboSuite 5.5 properly to do the data extraction
Solution: Research user manuals and contact tech support for additional help.

Problem: Standardizing the data.
Solution: Decide on a format and manually go through extracted data and determine what needs to be standardized.

Problem: Writing accurate and detailed pseudocode without having fully researched the technologies and languages that will be used (JSP, JDBC, JAVA, etc.)
Solution: Further research the technologies and starting the programming phase of this project.

Problem: Checking that data extracted by RoboSuite 5.5 was properly entered into the OlympicsDB.
Solution: Check by hand against the source.

Problem: Learning JavaBeans to interact with JSP
Solution: Research and follow examples of JavaBeans

Problem: Understanding JSP enough to access the JavaBean and Java code
Solution: Research and follow examples

Problem: Getting the results back to the Client and formatting them.
Solution: Research and follow examples

Problem: Building the SQL queries
Solution: Research and follow examples

2.5.2 Conceptual Problems and Solutions

Problem: Locating the data
Solution: Research the web using google and other search engines.

Problem: Identifying a complete set of tasks at this phase of the project.
Solution: Deeper analysis of the enterprise; moving to the second phase of the project where the enterprise will be conceptually and logically emulated.

Problem: How to calculate Olympic and world records efficiently.
Solution: Use properties of SQL to use comparisons to derive these attributes.

Problem: Deriving the first year a country participated.
Solution: Use properties of SQL to use comparison to find this information.

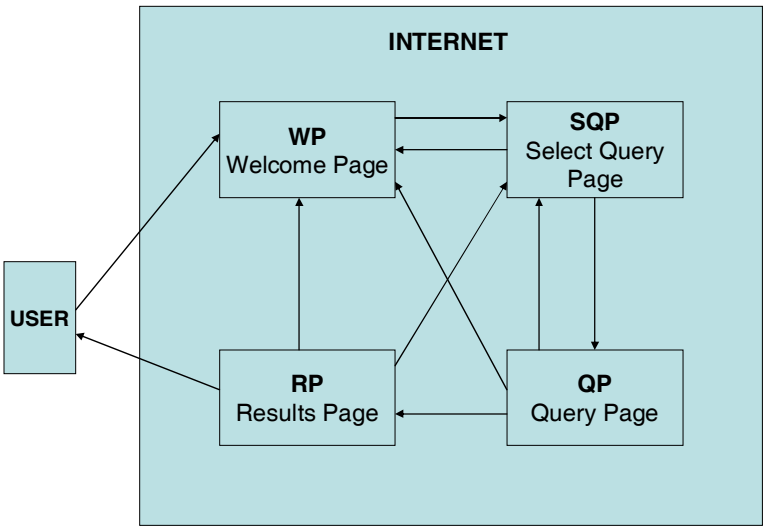
3. SYSTEM ANALYSIS AND SPECIFICATION

3.1 Description of Procedures

OlympiChronicles operates via a web browser that allows users to select various search criteria in researching facts about summer Olympic Games from 1896 through the present.

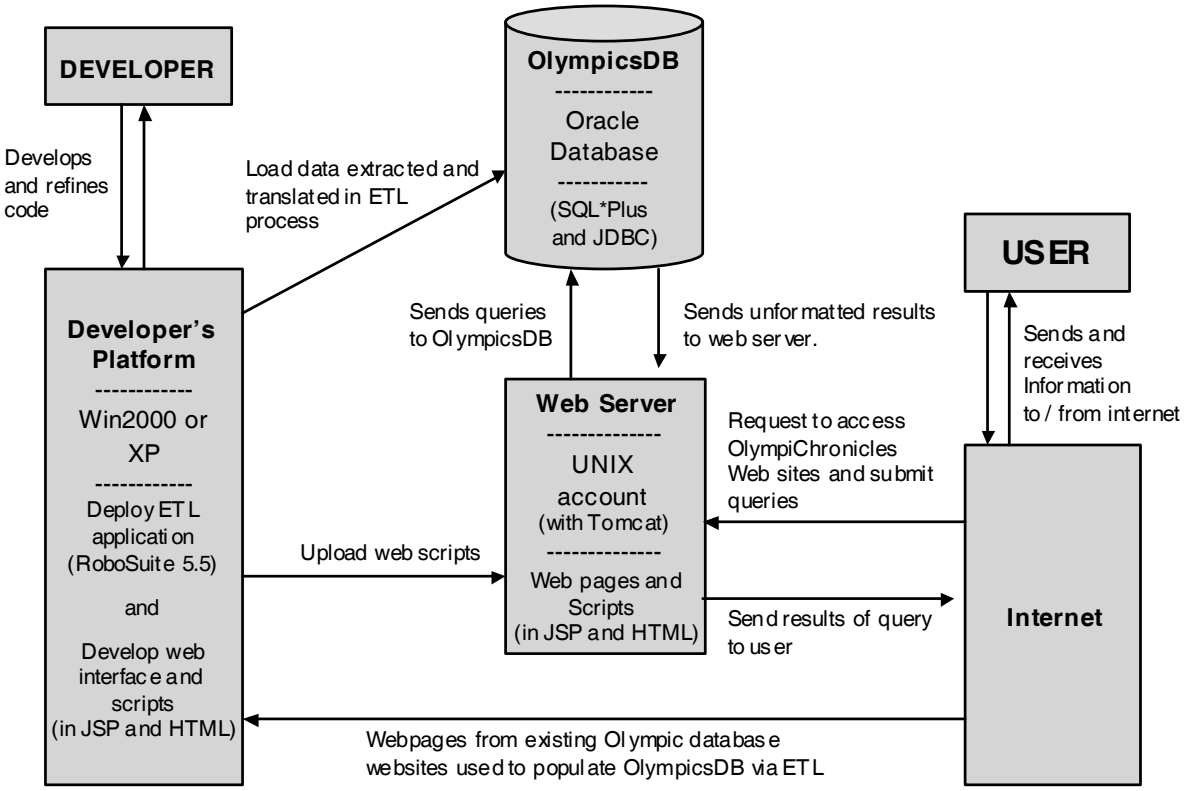
3.1.1 From the user's perspective:

The first step in accessing OlympiChronicles powerful database is to navigate to a predefined website. There the user will create a query and submit it to a process running on a remote server where the OlympicsDB is stored. This process will create a form containing SQL commands for the specified query and will submit it to the database. After the data has been retrieved from the database it will be formatted and presented through the user's web browser.



3.1.2 From the developer's perspective:

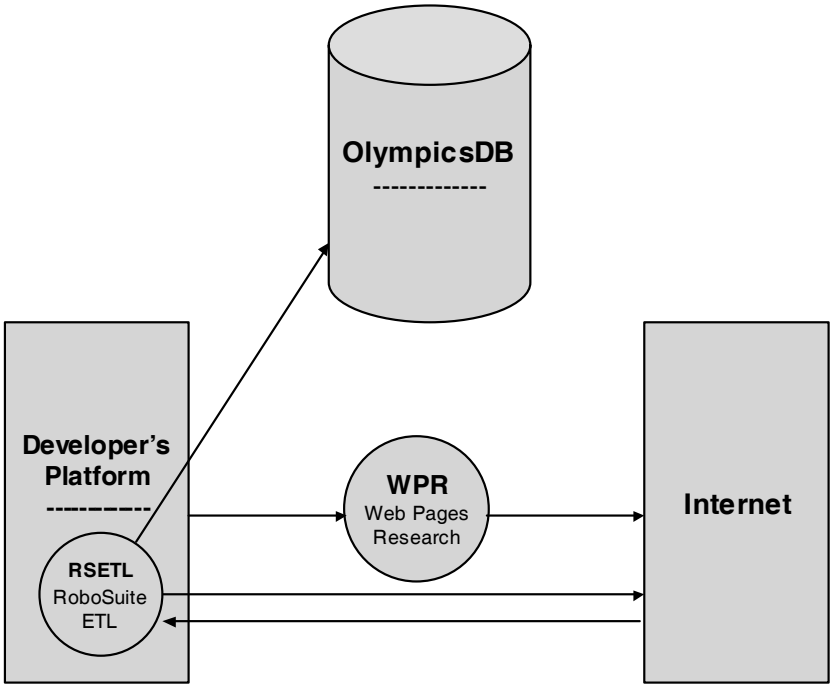
The developers will be employing several technologies in order to implement the enterprise in its varying phases. The diagram below shows the main components of the system and indicates what responsibility to the system each component has as well as the general flow of information. Parts of this diagram will be elaborated upon in subsequent sections.



3.1.3 ETL Procedures

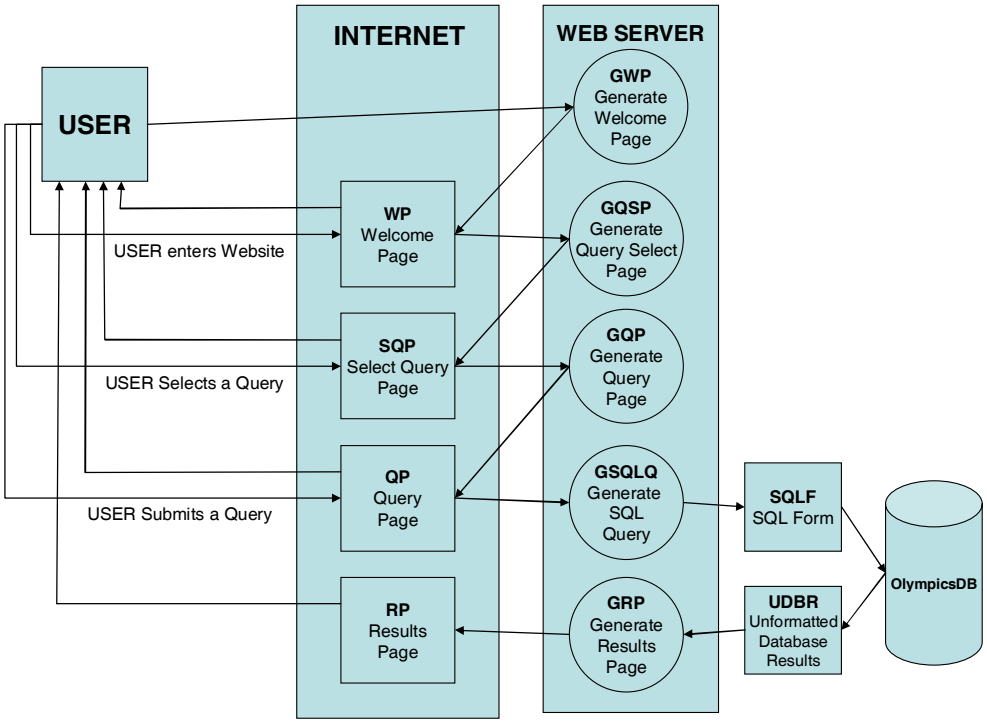
The **dBoss Company** designers will research, analyze, and select the most relevant summer Olympics websites. With the resulting websites bookmarked, the Kapow RoboSuite 5.5 ETL tool will be programmed to automatically surf

to the various websites, query the relevant data, extract it from the resulting tables, transform it into the required format, and load it into the OlympicsDB tables located on the Oracle server to be used by **OlympiChronicles** to answer the different user queries through a web interface.



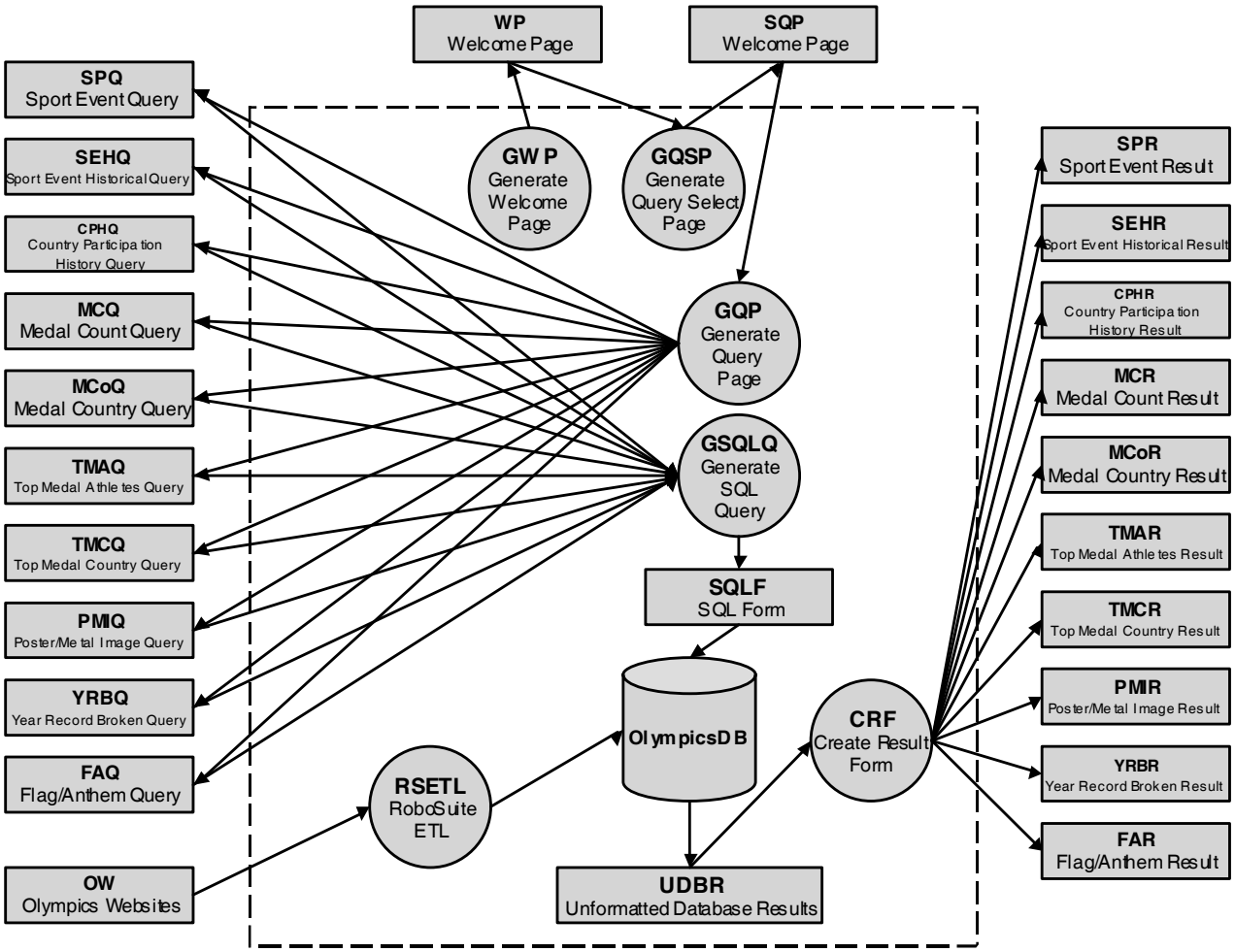
3.1.4 Web Server Procedures

The **OlympiChronicles** internal procedures include the engine that powers the website. This involves providing various queries and results of those queries available to the user. This is accomplished by several scripts and code for pages located on a UNIX web server. When a user navigates to the OlympiChronicles website, the initial web page is generated and served to the user. The remaining scripts and procedures will be described by following a typical use case scenario. The user will proceed to enter the website. A page with a list of ten queries is presented to the user. The user will then select one from among the queries to be performed. The selection will be sent to the web server where another procedure will generate a page with various options relevant to that query. The user will fill out the options desired and submit the query. The web server will receive the query request; generate the appropriate SQL commands which are then sent to the OlympicsDB. The database will then produce a results table and send it to the server from which the query was sent. Another process on the server will format the results into a web page and serve it up to the user.



3.2 Documentation

3.2.1 Top-Level Flow Diagram



3.2.2 Tasks, Subtasks, and Task Forms

3.2.2.1 Web pages Research Task

TASK NUMBER:	WPRT
TASK NAME:	Web Pages Research
PERFORMER:	OlympiChronicles Designers
PURPOSE:	To research the internet for web sites that contain data for the summer Olympic Games since 1896 until the present.
ENABLING COND:	To populate the OlympicsDB.
DESCRIPTION:	Research the internet
FREQUENCY:	As often as necessary
DURATION:	Varies
IMPORTANCE:	Critical
MAXIMUM DELAY:	N/A
INPUT:	Web queries
OUTPUT:	Index of queried results
DOCUMENT USE:	Web based search engines
OPS PERFORMED:	Researching and bookmarking web sites and/or pages with summer Olympic game data
SUBTASKS:	None
ERROR COND:	None

3.2.2.2 ETL Task

TASK NUMBER:	ETLT
TASK NAME:	Extract, Transform, and Load Task
PERFORMER:	Kapow RoboSuite 5.5
PURPOSE:	To extract data, transform or reformat it and load it into the OlympicsDB
ENABLING COND:	The creation of the OlympicsDB and any addition of data or updates to the OlympicsDB.
DESCRIPTION:	This tool (Kapow RoboSuite 5.5) extracts specific data from a web page, and load it into a predefined data relation or table.
FREQUENCY:	Once for the creation of the OlympicsDB and during any updates.
DURATION:	Varies
IMPORTANCE:	Critical
MAXIMUM DELAY:	N/A
INPUT:	A selected web page
OUTPUT:	Data into a relation in the OlympicsDB
DOCUMENT USE:	HTML documents
OPS PERFORMED:	Data extraction, data transformation, and data loading.
SUBTASKS:	Web pages Research
ERROR COND:	None

3.2.2.3 Generate Welcome Page Task

TASK NUMBER:	GWPT
TASK NAME:	Generate Welcome Page
PERFORMER:	Apache/Tomcat web server
PURPOSE:	To generate the welcome page.
ENABLING COND:	User accessing the OlympiChronicles web interface.
DESCRIPTION:	The Apache/Tomcat web server will generate the OlympiChronicles welcome page (home page) when a user wants to access the information on the OlympicsDB.
FREQUENCY:	As often as a user accesses the web address
DURATION:	Very short
IMPORTANCE:	Critical
MAXIMUM DELAY:	10 seconds
INPUT:	None
OUTPUT:	Welcome Page
DOCUMENT USE:	WIFWF: Web Interface Welcome Form
OPS PERFORMED:	Generation of welcome page, send it to the user and wait for user action.
SUBTASKS:	None
ERROR COND:	If A/TServer == busy, then Process=TimeOut.

3.2.2.4 Generate Query Select Page Task

TASK NUMBER:	GQSPT
TASK NAME:	Generate Query Select
PERFORMER:	Apache/Tomcat web server
PURPOSE:	To generate the query select page.
ENABLING COND:	User clicking on the ENTER button in the Welcome Page.
DESCRIPTION:	The Apache/Tomcat web server will generate the OlympiChronicles query-select page when requested by the user (from the Welcome page)
FREQUENCY:	As often as the user clicks on the ENTER button on the Welcome Page or the BACK button on a Query Page.
DURATION:	Very short
IMPORTANCE:	Critical
MAXIMUM DELAY:	10 seconds
INPUT:	Signal request from user to web server.
OUTPUT:	Query Select Page
DOCUMENT USE:	WISF: Web Interface Select Form
OPS PERFORMED:	Generate the Query Select page, send it to the user and wait for user input.
SUBTASKS:	None
ERROR COND:	If A/TServer == busy, then Process=TimeOut.

3.2.2.5 Generate Query Page Task

TASK NUMBER:	GQPT
TASK NAME:	Generate Query Page
PERFORMER:	Apache/Tomcat web server
PURPOSE:	To generate the query page where users will select their options for a specific query.
ENABLING COND:	Selecting a query on the Query Select Page
DESCRIPTION:	The Apache/Tomcat web server will generate the Query page every time a user selects a query from the Query Select Page.
FREQUENCY:	As often as the user clicks on a query to select it from the Query Select Page.
DURATION:	Very short
IMPORTANCE:	Critical
MAXIMUM DELAY:	10 seconds
INPUT:	Signal request from user to web server.
OUTPUT:	Query Page
DOCUMENT USE:	WIQF: Web Interface Query Form
OPS PERFORMED:	Generate the Query From, send it to user, allow user to make selections, wait for user input (submit).
SUBTASKS:	None
ERROR COND:	If A/TServer == busy, then Process=TimeOut.

3.2.2.6 Generate SQL Query Task

TASK NUMBER:	SQLCT
TASK NAME:	SQL Creation
PERFORMER:	Apache/Tomcat web server
PURPOSE:	Create SQL commands
ENABLING COND:	Submitting web query form
DESCRIPTION:	Generation of SQL commands from a web query form to task the OlympicsDB.
FREQUENCY:	Once per user query submission.
DURATION:	Short
IMPORTANCE:	Critical
MAXIMUM DELAY:	5-10 seconds
INPUT:	Web query form
OUTPUT:	(SQLF) SQL Form
DOCUMENT USE:	(SPQ) Sport Event Query; (SEHQ) Sport Event Historical Query; (CPHQ) Country Participation History Query; (MCQ) Medal Count Query; (MCoQ) Medal Country Query; (TMAQ) Top Medal Athletes Query; (TMCQ) Top Medal Country Query; (PMIQ) Poster/Medal Image Query; (YRBQ) Year Record Broken Query, or (FAQ) Flag/Anthem Query.
OPS PERFORMED:	if Q==SPQ || Q==SEHQ || Q==CPHQ || Q==MCQ || Q==MCoQ || Q==TMAQ || Q==TMCQ || Q==PMIQ, then create SQL commands, else report error.
SUBTASKS:	None
ERROR COND:	If A/TServer == busy, then Process=TimeOut. See OPS PERFORMED

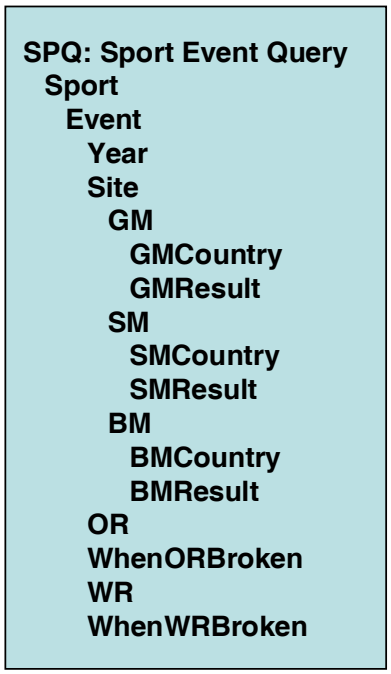
3.2.2.7 Generate Results Page Task

TASK NUMBER:	GRPT
TASK NAME:	Generate Results Page
PERFORMER:	Apache/Tomcat web server
PURPOSE:	To generate the results page.
ENABLING COND:	Getting the Query Result Form, after querying the OlympicsDB.
DESCRIPTION:	The Apache/Tomcat web server will generate the Results Page to be displayed to the user, after getting the Query Result Form.
FREQUENCY:	As often as the user submits a query and the query is valid.
DURATION:	Short
IMPORTANCE:	Critical
MAXIMUM DELAY:	10 seconds
INPUT:	Query Result Form
OUTPUT:	Results Page
DOCUMENT USE:	WIRF: Web Interface Result Form
OPS PERFORMED:	Generate a Results Page to be displayed to the user from the Result form.
SUBTASKS:	None
ERROR COND:	If A/TServer == busy, then Process=TimeOut.

3.2.2.8 Create Query Result Form Task

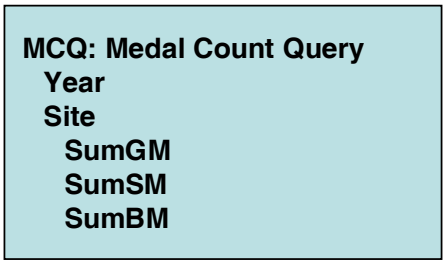
TASK NUMBER:	CRFT
TASK NAME:	Create Result Form
PERFORMER:	Server side script
PURPOSE:	Provide a formatted result from the OlympicsDB.
ENABLING COND:	Database completing operations.
DESCRIPTION:	Formats output of the extracted data from the OlympicsDB to a form that can be interpreted by a web browser.
FREQUENCY:	Once per user query submission.
DURATION:	Depends on the complexity of the query result.
IMPORTANCE:	Critical
MAXIMUM DELAY:	5-10 seconds
INPUT:	OlympicsDB data
OUTPUT:	(SPR) Sport Event Result; (SEHR) Sport Event Historical Result; (CPHR) Country Participation History Result; (MCR) Medal Count Result; (MCoR) Medal Country Result; (TMAR) Top Medal Athletes Result; (TMCR) Top Medal Country Result; (PMIR) Poster/Medal Image Result; (YRBR) Year Record Broken Result, or (FAR) Flag/Anthem Result.
DOCUMENT USE:	None
OPS PERFORMED:	Transform data from the OlympicsDB output format to a web browser compatible format.
SUBTASKS:	None
ERROR COND:	If OlympicsDB_output=unknown, then produce error message and stop.

3.2.3 Document Forms

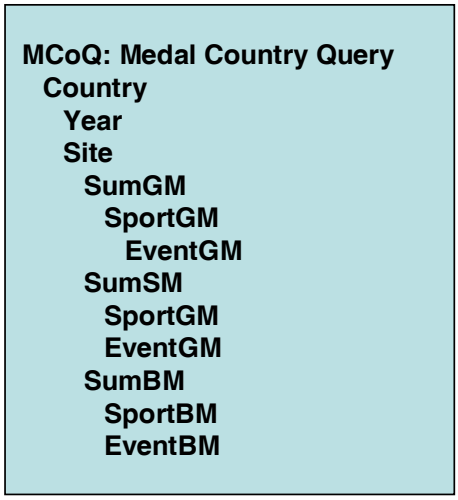

SPQ: Sport Event Query
Sport
Event
Year
Site
GM
GMCountry
GMResult
SM
SMCountry
SMResult
BM
BMCountry
BMResult
OR
WhenORBroken
WR
WhenWRBroken

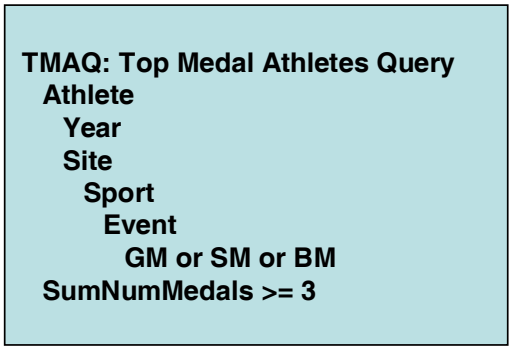

SEHQ: Sport Event History Query
Sport
Event
Year
Site
GM
GMCountry
GMResult
SM
SMCountry
SMResult
BM
BMCountry
BMResult

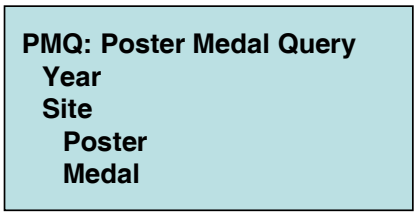

CPHQ: Country Participation History Query
Country
YearFirstParticipated
Year
Site
Sport
Event
SumNumGames

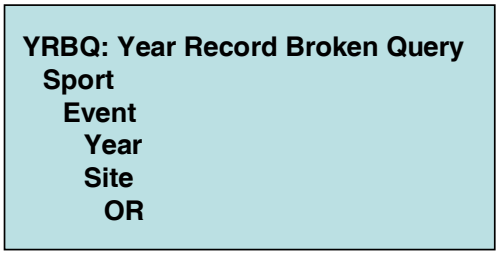

MCQ: Medal Count Query
Year
Site
SumGM
SumSM
SumBM

3.2.3 Document Forms Continued

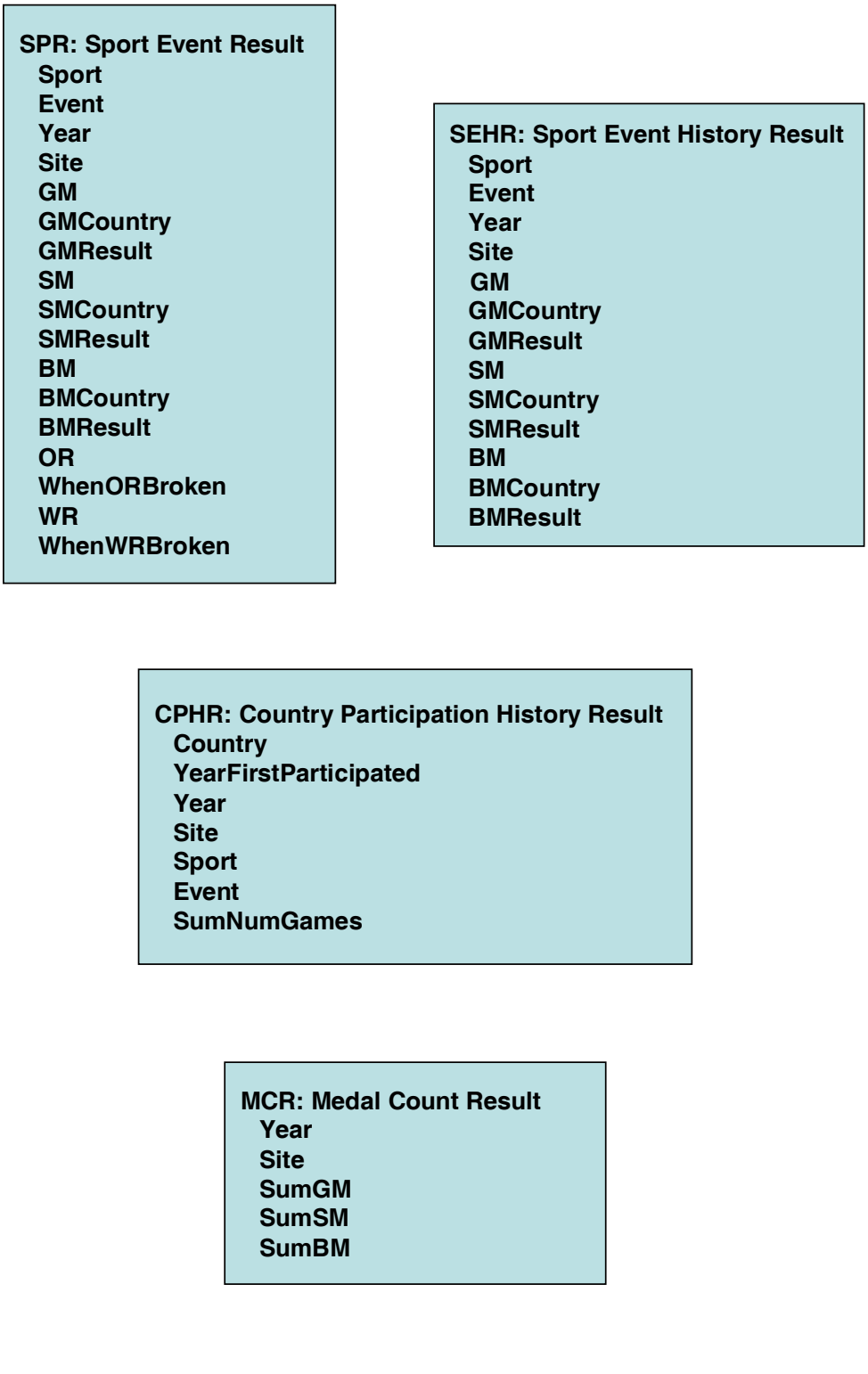




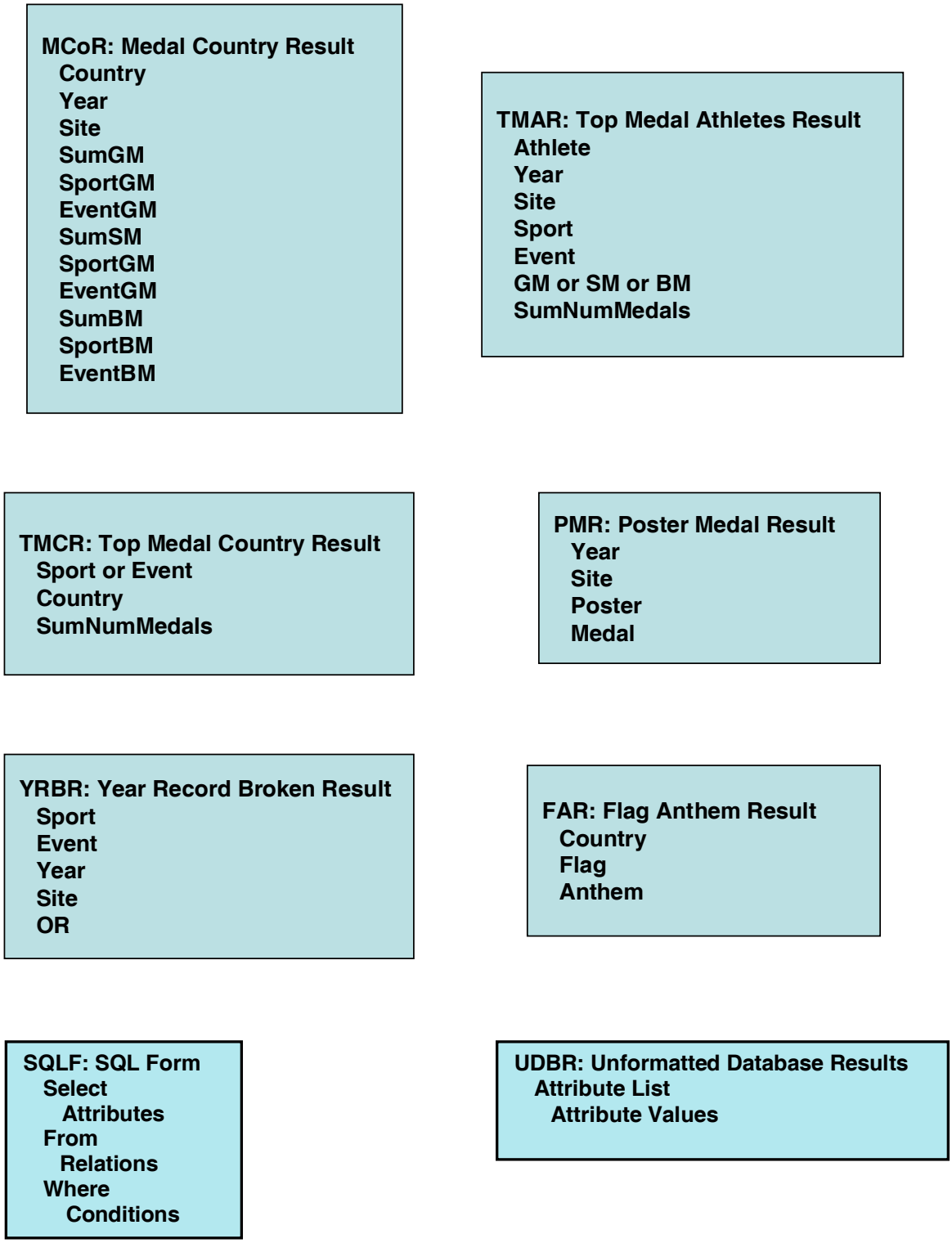




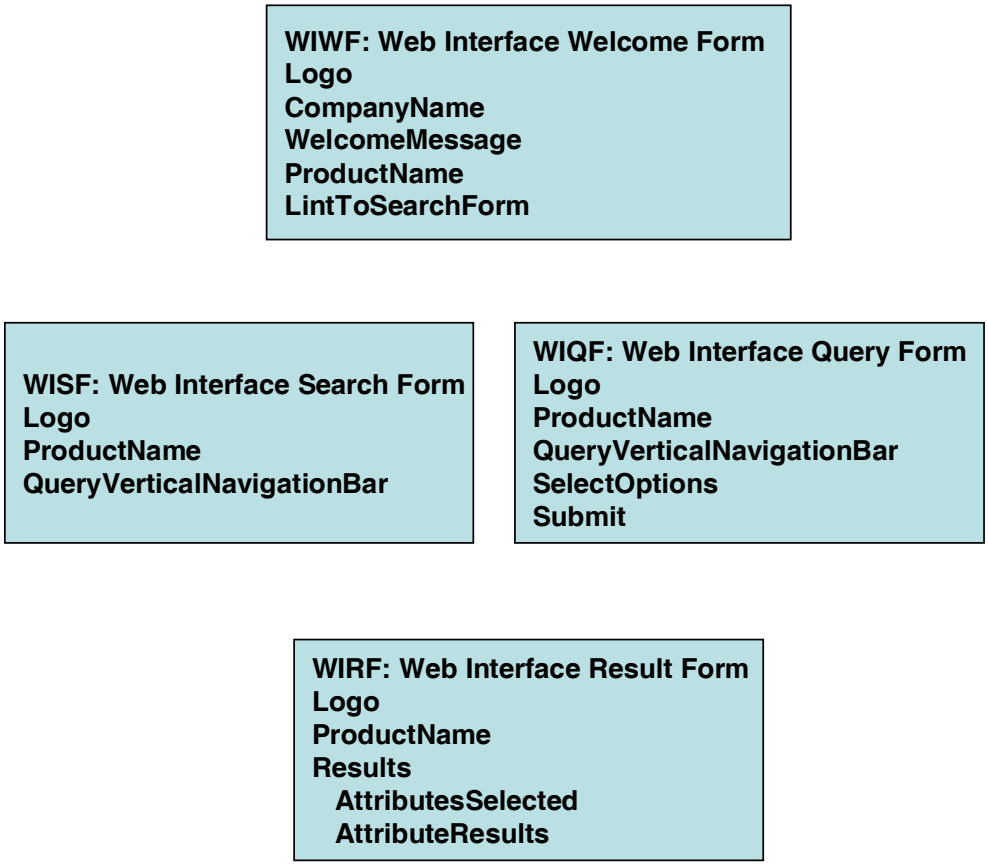
3.2.3 Document Forms Continued



3.2.3 Document Forms Continued



3.2.3 Document Forms Continued

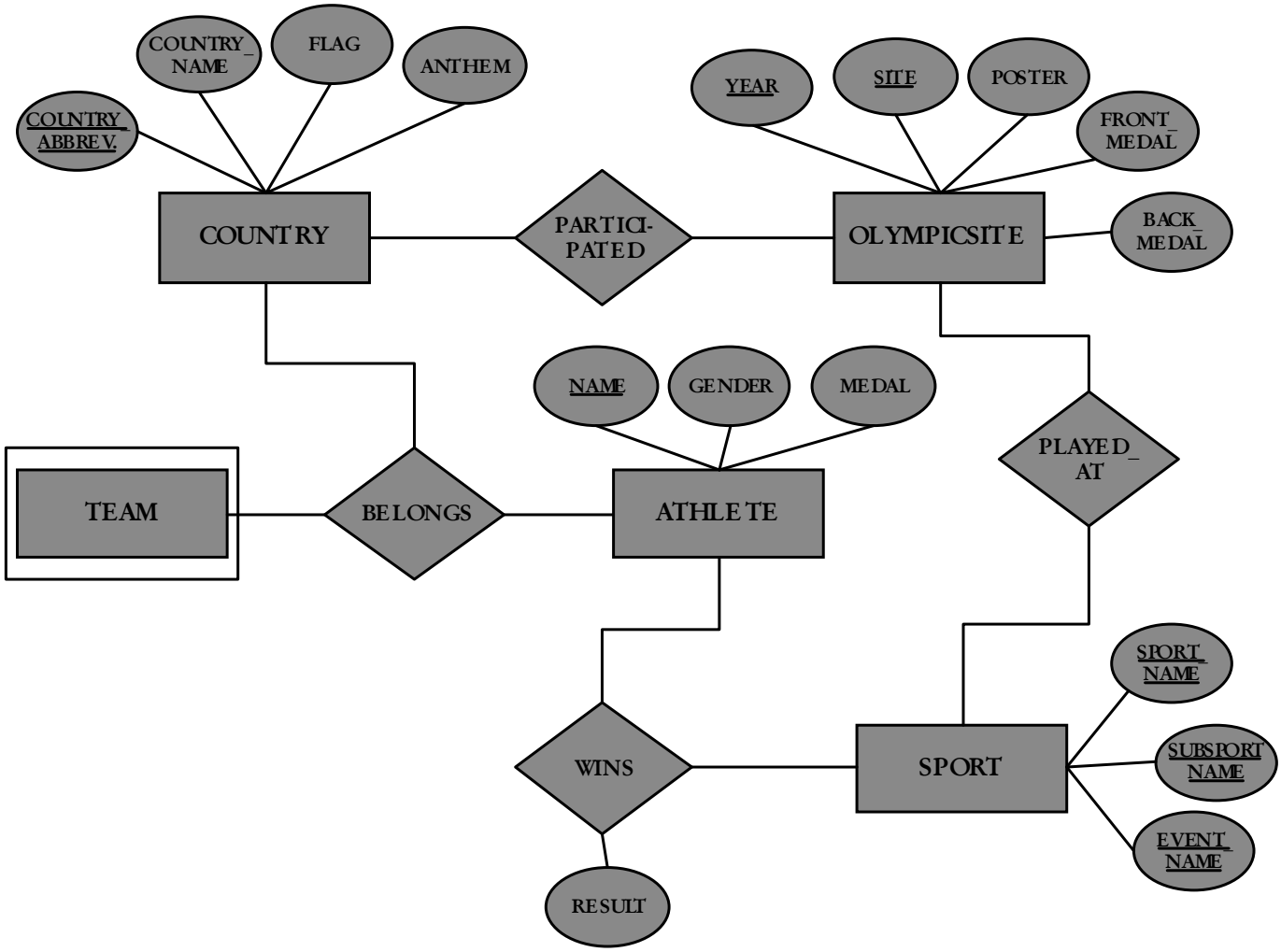


4 CONCEPTUAL MODELING

4.1 Conceptual Schema

The conceptual schema is the higher level representation of the **OlympiChronicles** enterprise as conceived by the designers. This includes the different identified entities and relationships based on the document forms specified in the Requirements Document for this enterprise. This entities and relationships include the internal processes by which the data will be extracted, transformed, and loaded (ETL) into the OlympicsDB, as well as the process of user queries.

4.1.1 ER Model Graphical Schema



4.2 Functional Dependencies

The functional dependencies identified are:
For Country entity:

- *Country_Abbreviation* → *Country_Name*
- *Country_Abbreviation* → *Flag*
- *Country_Abbreviation* → *Anthem*
- *Country_Abbreviation* → *First_Year_Participated*

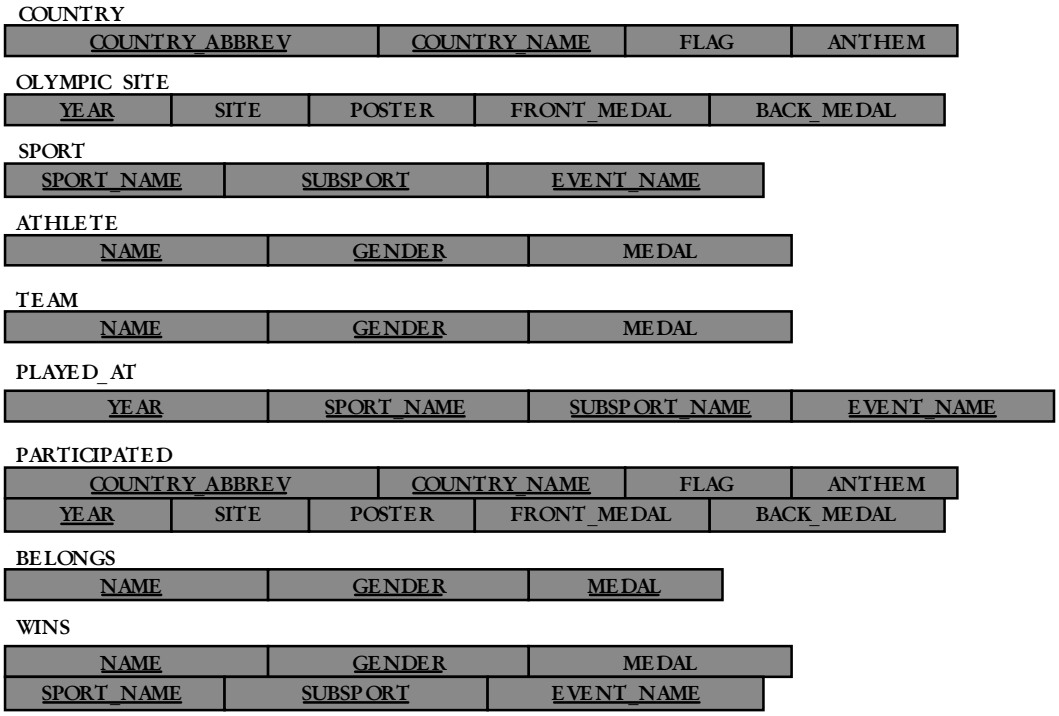
For OlympicSite entity:

- *Year* → *Site*
- *Year* → *Poster*
- *Year* → *Medal*

5 LOGICAL MODELING

5.1 Logical Schema

The logical schema is the next level in the representation of the **OlympiChronicles** enterprise comprised of the relation schemas derived from the ER diagram in the conceptual schema.



5.1.1 Relational Model
5.1.2 Normalization

Relations need to be in either Boyce-Codd normal form (BCNF) or in Third normal form (3NF) in order to obtain lossless and sometimes dependency preserving relations. In order to normalize these relations, we need to use the functional dependencies derived in the previous section, and check if the relations are BCNF or 3NF, and if they are not, then the relations need to be decomposed into BCNF or 3NF relations.

6 TASK EMULATION

6.1 Task Design Specification

6.1.1 Extract, Transform, and Load Task

6.1.1.1 Web Pages Research Task

Extract, Transform, and Load

```
Start RoboSuite 5.5
Configure RoboSuite 5.5
        for each website bookmarked
                for each webpage on website [query results]
                        RoboSuite.url = webpage.url
                        set values to look for
                        extract information to a predefined
                        table.
```

Web Pages Research

```
{Google query to find Summer Olympic Games sites}
For each website found in Google
        if website has relevant data and if website has complete
           data to be used by the OlympicsDB
                Bookmark
        else
                skip
```

6.1.2 Generate Welcome Page Task

6.1.3 Generate Query Select Page Task

```
Generate Welcome Page
{HTML for webpage layout}
{HTML for OlympiChronicles logo}
{HTML for Company Logo}
If click_to_enter == true
        link to GQSP (Query Select Page)
Else
        no action
```

```
Generate Query Select
{HTML for webpage layout}
{HTML for OlympiChronicles logo}
{HTML for header "Summer Olympics Facts!"}
Intro_text ="Place your mouse over a query to see more information"
{HTML/JavaScript for query options vertical navigation bar}
//general code for mouse over any query
If click_logo == true
        link to this (Query Select Page)
If mouse_over_query
        display information and link to go to query
        if query_info_displayed
                if click_to_select_query == true
                        link to GQPT (Query Page)
                else
                        no action
```

6.1.4 Generate Query Page Task

Generate Query Page

```
{HTML for OlympiChronicles logo}
{HTML to display Query_Name as a header/title for webpage}
{HTML to display the options users have to select for a specific Query}
//general code for selection for any query
If sport_related query
        display choose_olympic_game
        display choose_sport
        display choose_event
        //other info depending on query
If country_related query
        display choose_olympic_game
        display choose country
If athlete_related query
        display choose_olympic_game
//after choosing options
If click_submit == true
        link to SQL Query (Generate SQL Query)
        query OlympicsDB
        getResults (from OlympicsDB)
        link to GRP (Results Page)
Else if click_logo == true
        link to GSQP (Select Query Page)
Else
        no action
```

6.1.5 Generate SQL Query Task

Generate SQL

```
If query == Sport_Event_Query
    SELECT year, site, sport_name, subsport_name, event_name,
        subevent_name, medal
    FROM Sports, OlympicSites, Medal, Wins, Played_At
    WHERE year=year_chosen and site=site_chosen and
        sport_name=sport_name_chosen and
        subsport_name=subsport_name_chosen and
        event_name=event_name_chosen and
        subevent_name=subevent_name_chosen and
        medal=medal_chosen
Else if query == Sport_Event_Historical_Query
    SELECT year, site, sport_name, subsport_name, event_name
        subevent_name, medal
    FROM Sports, OlympicSites, Medal
Else if query == Country_Participation_History_Query
    SELECT C.year, site, country_abbreviation, C.country_name,
        year_first_participated, count(country_name)
    FROM Country C, OlympicSite O, Participated P
    GROUP BY P.year
Else if query == Medal_Count_Query
    SELECT year, site, country_name, count(medal)
    FROM OlympicSite, Country, Medal, Participated,
        Wins, Belongs
    WHERE year=year_chosen and site=site_chosen and
        country_name=country_name_chosen
    GROUP BY medal
Else if query == Medal Country History Query
    SELECT year, site, country_name, medal
    FROM OlympicSite, Country, Medal, Participated, Wins
        Belongs
```

Generate SQL (cont...)

```
Else if query == Top_Medal_Athletes_Query
        SELECT year, site, first_name, last_name, medal
        FROM OlympicSite, Athlete, Belongs, Participated, Medal,
                Sport, Wins, Played_At
        HAVING count (medal) >3
Else if query == Top_Medal_Country_Query
        SELECT year, site, country_name, event_name, count(medal)
        FROM OlympicSite, Country, Sport, Medal, Win, Participated
                Played_At
Else if query == Poster/Medal_Image_Query
        SELECT year, site, poster, front_medal, back_medal
        FROM OlympicSite
        WHERE year=year_chosen and site=site_chosen
Else if query == Year_Record_Broken_Query
        SELECT
        FROM
        WHERE
Else if query == Flag/Anthem_Query
        SELECT year, site, country_name, flag, anthem
        FROM OlympicSite, Country
        WHERE year=year_chosen and site=site_chosen and
                country_name=country_name_chosen
```

6.1.6 Generate Result Page Task

```
Generate Results Page
{HTML for webpage layout}
{HTML for OlympiChronicles logo}
{HTML for header – name of query}
//general for any query result
Generate table
        get (results_table)
        display (results_table)
If click_logo == true
        link to GQSP (Query Select Page)
```

6.1.7 Create Result Form Task

Create Result From

```
Results_table.create_table (2 columns)
Results_table.add_columns ("Year", "Site")
If result == Sport_Event_Result || result == Sport_Event_History_
                                                    Result
        results_table.add_columns ("Sport", "Subsport", "Event",
                "Subevent", "Medals")
        populate with results from OlympicsDB
Else if result == Country_Participation_History_Result
        results_table.add_columns ("Country Abbreviation",
                "Country", "First Year Country Participated",
                "Total Number Games")
        populate with results from OlympicsDB
Else if result == Medal_Count_Result
        results_table.add_columns ("Country", "Total Number of
                Medal")
         populate with results from OlympicsDB
Else if result == Medal_Country_Result
        results_table.add_columns ("Country", "Gold Medals",
                "Silver Medals", "Bronze Medals")
        populate with results from OlympicsDB
Else if result == Top_Medal_Athletes_Result
        results_table.add_columns ("Team" or "Athlete Name",
                "Medals")
         populate with results from OlympicsDB
Else if result == Top_Medal_Country_Result
        results_table.add_columns ("Country", "Event", "Total
                Number of Medals")
         populate with results from OlympicsDB
```

Create Result From (cont...)

```
If result == Poster/Medal Image Result
        result_table.add_columns ("Poster url", "Front Medal url",
                Back Medal url")
         populate with results from OlympicsDB
Else if result == Year_Record_Broken_Result
        result_table.add_columns ("")
         populate with results from OlympicsDB
Else if result == Flag/Anthem_Result
        result_table.add_column ("Country", "Flag url", "Anthem
                url")
         populate with results from OlympicsDB

Send result_table to GRP (Results Page)
```

7 SOURCE PROGRAM LISTING

7.1 Web Interface

INDEX.HTML (Welcome Page)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>OlympiChronicles Welcome Page</title>
<script language="JavaScript" type="text/JavaScript">
<!--
function MM_reloadPage(init) {  //reloads the window if Nav4 resized
  if (init==true) with (navigator) {if
((appName=="Netscape")&&(parseInt(appVersion)==4)) {
    document.MM_pgW=innerWidth; document.MM_pgH=innerHeight;
onresize=MM_reloadPage; }}
  else if (innerWidth!=document.MM_pgW || innerHeight!=document.MM_pgH)
location.reload();
}
MM_reloadPage(true);

function MM_preloadImages() { //v3.0
  var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
    var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length;
i++)
    if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
}

function MM_swapImgRestore() { //v3.0
  var i,x,a=document.MM_sr; for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++)
x.src=x.oSrc;
}

function MM_findObj(n, d) { //v4.01
  var p,i,x;  if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
    d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++)
x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++)
x=MM_findObj(n,d.layers[i].document);
  if(!x && d.getElementById) x=d.getElementById(n); return x;
}

function MM_swapImage() { //v3.0
```

INDEX.HTML (Welcome Page Continued...)

```
//-->
</script>
<style type="text/css">
<!--
.style2 {font-size: 18px}
.style3 {font-size: small}
-->
</style></head>

<body onLoad="MM_preloadImages('images/lighted_torch.gif')">
<div align="center">
  <p class="style3">This site has been optimized for Internet Explorer 4.0 or
greater.<br>
    <a
href="http://www.microsoft.com/windows/ie/downloads/critical/ie6sp1/default.
mspx">Get I\
t Here (Internet Explorer 6 SP 1)!!! </a></p>
  <p> </p>
  <p> </p>
  <p> </p>
  <p><a href="splashScreen.html" onMouseOut="MM_swapImgRestore()"
onMouseOver="MM_swapImage('T\
orch','','images/lighted_torch.gif',1)"><img src="images/unlighted_torch.gif"
alt="CLICK to go\
 to Query Select Page" name="Torch" width="87" height="218"
border="0"></a></p>
  <p> </p>
  <p><em><span class="style2">Light the <strong>TORCH</strong>
above to enter OlympiChronicles\
```

SPLASHSCREEN.HTML (Redirection Page)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta http-equiv="refresh" content="10;URL=QuerySelect.html">
<title>Untitled Document</title>
<style type="text/css">
<!--
.style1 {font-size: large}
```

SPLASHSCREEN.HTML (Redirection Page Continued...)

```
.style6 {font-size: x-large; font-family: "Berlin Sans FB Demi";}
.style8 {color: #000000}
.style9 {color: #FF0000}
-->
</style>
</head>

<body>
<div align="center">
  <p> </p>
  <p><img src="images/torch_1.gif" width="50" height="100"><a
href="QuerySelect.html"><img src\
="images/BrandLogoWebNEW.jpg" width="463" height="290"
border="0"></a><img src="images/torch_1\
.gif" width="50" height="100"></p>
  <hr>
  <p class="style6">
    Another internet resource for information on the summer Olympic games<br>
from 1896 to present.</p>
  <hr>
  <p class="style1"> </p>
  <p class="style2">  <span class="style3"><blink><span class="style8">You
will be <span class=\
"style9">transferred automatically</span> to the Welcome Page <span
class="style9">in 10 secon\
ds</span>. <br>
    If your browser does not support automatic forwarding, please click on the above
Logo.</sp\
an></blink></span><br>
    <br>
    </p>
  <p>Developed by...<br>
      <img src="images/LogoWeb.jpg" width="214" height="174"></p>
 Steffanie <span class="ctn">Orellana and Craig Shapiro </span>
  <p class="style2"><br>
  </p>
</div>
</body>
</html>
```

QUERYSELECT.HTML (Query Select Page ...)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<title>OlympiChronicles Query Select ---> Please choose from one of the
queries.</title>

<script language="JavaScript" type="text/JavaScript">
<!--
function MM_preloadImages() { //v3.0
  var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
    var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length;
i++)
    if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
}

function MM_findObj(n, d) { //v4.01
  var p,i,x;  if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
    d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++)
x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++)
x=MM_findObj(n,d.layers[i].document);
  if(!x && d.getElementById) x=d.getElementById(n); return x;
}

function MM_nbGroup(event, grpName) { //v6.0
  var i,img,nbArr,args=MM_nbGroup.arguments;
  if (event == "init" && args.length > 2) {
    if ((img = MM_findObj(args[2])) != null && !img.MM_init) {
      img.MM_init = true; img.MM_up = args[3]; img.MM_dn = img.src;
      if ((nbArr = document[grpName]) == null) nbArr = document[grpName] =
new Array();
      nbArr[nbArr.length] = img;
      for (i=4; i < args.length-1; i+=2) if ((img = MM_findObj(args[i])) != null) {
        if (!img.MM_up) img.MM_up = img.src;
```

QUERYSELECT.HTML (Query Select Page Continued...)

```
    for (i=0; i < nbArr.length; i++) { img=nbArr[i]; img.src = img.MM_up; img.MM_dn
= 0; }
   document[grpName] = nbArr = new Array();
   for (i=2; i < args.length-1; i+=2) if ((img = MM_findObj(args[i])) != null) {
    if (!img.MM_up) img.MM_up = img.src;
    img.src = img.MM_dn = (args[i+1])? args[i+1] : img.MM_up;
    nbArr[nbArr.length] = img;
 } }
 //-->
 </script>
 <script language="JavaScript" type="text/JavaScript">
 <!--
 function MM_reloadPage(init) {  //reloads the window if Nav4 resized
   if (init==true) with (navigator) {if
((appName=="Netscape")&&(parseInt(appVersion)==4)) {
     document.MM_pgW=innerWidth; document.MM_pgH=innerHeight;
onresize=MM_reloadPage; }}
   else if (innerWidth!=document.MM_pgW || innerHeight!=document.MM_pgH)
location.reload();
 }
 MM_reloadPage(true);

 function MM_showHideLayers() { //v6.0
  var i,p,v,obj,args=MM_showHideLayers.arguments;
  for (i=0; i<(args.length-2); i+=3) if ((obj=MM_findObj(args[i]))!=null) { v=args[i+2];
    if (obj.style) { obj=obj.style; v=(v=='show')?'visible':(v=='hide')?'hidden':v; }
    obj.visibility=v; }
 }
 //-->
 </script>
 <style type="text/css">
 <!--
 .style3 {font-size: large}
 .style4 {font-size: 18px}
```

QUERYSELECT.HTML (Query Select Page Continued...)

```
    place your mouse pointer on one<br>
     of the ten catagories to the left. A description <br>
     of that query will appear in this window.</p>
     <p class="style3">When you have decided which catagory you would like <br>
      to research, click on the title and you will be <br>
      presented with a menu of options for<br>
      that query designed to assist you <br>
      in
      narrowing your results.</p>
     <p class="style3"> </p>
     <p class="style9"><span class="style8">Have Fun and ENJOY!</span></p>
   </div>
  </div>
  <div id="Layer6" style="position:absolute; left:304px; top:276px; width:491px;
height:305px; z\
  -index:11; background-color: #FFFFFF; layer-background-color: #FFFFFF; border: 0px
none #00000\
  0; visibility: hidden;">
    <div align="center">
     <p><strong><span class="style10">POSTERS and MEDALS<br>
          <span class="style4">QUERY</span></span></strong></p>
     <p> </p>
     <p>The result of this query</p>
     <p>displays images of the official Olympics</p>
     <p> <strong>poster</strong> and <strong>medal</strong> (front and back)
</p>
     <p>for the selected Olympic year.</p>
    </div>
  </div>
```

QUERYSEH.JSP (Sport Event Query Page)

```
  <jsp:useBean id="SEHData" class="SQLUtilities.SEHData" scope="session" />
<jsp:setProperty name="SEHData" property="*"/>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page import="SQLUtilities.Utilities" %>
<%@ page errorPage="myError.jsp?from=QuerySEH.jsp"  %>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<title>OlympiChronicles Query : SPORT EVENT HISTORY</title>

<script language="JavaScript" type="text/JavaScript">
<!-- hide from old browsers...

var info = new Array(<%= Utilities.getSubSportANDEvent2() %>);

/**********************************************************************
**/
function stringSplit ( string, delimiter ) {
   if ( string == null || string == "" ) {
      return null;
```

QUERYSEH.JSP (Sport Event Query Page Continued)

```
/*************************************************************************************
*******/
 function createMenus () {

    for ( var i=0; i < info.length; i++ ) {
       menu1[i] = stringSplit ( info[i], '*' );
       menu2[i] = stringSplit ( menu1[i][1], '|' );
    }

    var disciplines = document.myForm.discipline;
    var events = document.myForm.eventt;

    disciplines.length = menu1.length;
    events.length = menu2[0].length;
    for ( var i=0; i < menu1.length; i++ ) {
        disciplines.options[i].value  = menu1[i][0];
        disciplines.options[i].text   = menu1[i][0];
    }
    document.myForm.discipline.selected = 0;
```

QUERYSEH.JSP (Sport Event Query Page Continued...)

```
[some more automatically generated code...]
<div id="Layer1" style="position:absolute; left:300px; top:186px; width:500px;
height:397px; b\
order:1px solid #999999; z-index:1; visibility: visible;">
  <div align="center">
    <p> </p>
   <p>Please select the options for your query:</p>

<form name="myForm" action="QueryResultSEH.jsp" method="post"><p>
Discipline:  
<select name="discipline" size=1
onChange="updateMenus(this);MM_showHideLayers('Layer16','','s\
how')";>
 <option>
```

QUERYCPH.JSP (Country Participation History Query Page)

```
<jsp:useBean id="CPHData" class="SQLUtilities.CPHData" scope="session" />
<jsp:setProperty name="CPHData" property="*"/>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page import="SQLUtilities.Utilities" %>
<%@ page errorPage="myError.jsp?from=QueryCPH.jsp"  %>

<html>
<head>
```

QUERYMC.JSP (Medal Count Query Page)

```
<jsp:useBean id="user" class="SQLUtilities.UserData" scope="session"/>
<jsp:setProperty name="user" property="*"/>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page import="SQLUtilities.Utilities" %>
```

QUERYMC.JSP (Medal Count Query Page Continued...)

```
/*************************************************************************
****/
function createMenus () {

   for ( var i=0; i < info.length; i++ ) {
      menu1[i] = stringSplit ( info[i], '*' );
      menu2[i] = stringSplit ( menu1[i][1], '|' );
   }

   var countries = document.myForm.select2;
   var years = document.myForm.countryyear;

   countries.length = menu1.length;
   years.length = menu2[0].length;
   for ( var i=0; i < menu1.length; i++ ) {
       countries.options[i].value  = menu1[i][0];
       countries.options[i].text   = menu1[i][0];
   }
   document.myForm.select2.selected = 0;
   for (var x=0; x < menu2[0].length; x++) {
       years.options[x].text = menu2[0][x];
       years.options[x].value = menu2[0][x];
   }
   document.myForm.countryyear.selected = 0;
}
/*************************************************************************
******/
function updateMenus ( what ) {
   var sel = what.selectedIndex;

   if ( sel >= 0 && sel < menu1.length )
      var temp = menu2[sel];
   else
      var temp = new Array ();

   what.form.countryyear.length = temp.length;

   for ( var i = 0; i < temp.length; i++ ) {
      what.form.countryyear.options[i].text  = temp[i];
      what.form.countryyear.options[i].value = temp[i];
   }
   what.form.countryyear.selected=0;
}
```

QUERYMC.JSP (Medal Count Query Page Continued...)

```
<div id="Layer1" style="position:absolute; left:300px; top:186px; width:500px;
height:397px; border:1px solid #999999; z-index:1; visibility: visible;">
  <div align="center">
    <p> </p>
   <p>Please select the options for your query:</p>
   <form name="myForm" action="QueryResultMC.jsp" method="post"><p>
      Country:  
     <select name="select2" size=1
onChange="updateMenus(this);MM_showHideLayers('Layer16','','show')">
 <option>   
      <option>
      <option>
     </select>
<p>
 <div id="Layer16" style="position:absolute; left:8px; top:116px; width:485px;
height:91px; z-index:2; visibility: hidden;">
Year: 
        <select name="countryyear" size=1>
      <option> 
      <option>
      <option>
  </select>
     <jsp:setProperty name="user" property="select2"
value="document.myForm.select2.options\
[document.myForm.select2.selectedIndex].text" />
      <jsp:setProperty name="user" property="countryyear"
value="document.myForm.countryyear\
.options[document.myForm.countryyear.selectedIndex].text" />
      <p> </p>
   <p> </p>
   <p> </p>
   <p> </p>
      <input type="submit" name="Submit" value="Submit">
      <input type="reset"  value="Reset">
      </div>
      </form>
  </div>
</div>
<div align="left">
</div>
<div align="left">
  <p>   <br>
  </p>
```

QUERYMpC.JSP (Medals per Count Query Page)

```
<jsp:useBean id="MPCData" class="SQLUtilities.MPCData" scope="session"/>
<jsp:setProperty name="MPCData" property="*"/>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page import="SQLUtilities.Utilities" %>
<%@ page errorPage="myError.jsp?from=Query.jsp"  %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Temp</title>

[automatically generated code...]

   <p>Please select the options for your query:</p>
   <form action="QueryResultMpC.jsp" method="post" name="form1" >

     <select name="year" onChange="MM_showHideLayers('Layer16','','show')">
       <option value="0" selected>Select Olympic Year</option>
       <%= Utilities.getYears() %>
     </select>
       <div id="Layer16" style="position:absolute; left:8px; top:116px; width:485px;
height:9\
1px; z-index:2; visibility: hidden;">

       <p> </p>
   <p> </p>
   <p> </p>
   <p> </p>

       <input type="submit" name="Submit" value="Submit">
       <input type="reset"  value="Reset">
       </div>

       </form>
  </div>
</div>
<div align="left">
</div>
```

QUERYPM.JSP (Poster/Medal Query Page)

```
<jsp:useBean id="user" class="SQLUtilities.UserData" scope="session"/>
<jsp:setProperty name="user" property="*"/>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page import="SQLUtilities.Utilities" %>
<%@ page errorPage="myError.jsp?from=QueryPM.jsp"  %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>OlympiChronicles Query : POSTERS and MEDALS</title>

[automatically generated code...]

   <p>Please select the options for your query:</p>
   <form action="QueryResultPM.jsp" method="post" name="form1" >
    <select name="select" onChange="MM_showHideLayers('Layer16','','show')">
     <option value="0" selected>Select Olympic Year</option>
     <%= Utilities.getYears() %>
    </select>
     <div id="Layer16" style="position:absolute; left:8px; top:116px; width:485px;
height:9\
1px; z-index:2; visibility: hidden;">
     <p>Choose from the options below:</p>
   <p>Poster      <input name="posterMedal" type="checkbox" value="poster">
     Medal - front   <input name="posterMedal" type="checkbox"
value="medalFront">
     Medal - back    <input name="posterMedal" type="checkbox"
value="medalBack">
     </p>
     <p> </p>
   <p> </p>
        <%
            if (!user.isCbValid ())
            {
```

QUERYFA.JSP (Flags/Anthems Query Page)

```
<jsp:useBean id="user" class="SQLUtilities.UserData" scope="session"/>
<jsp:setProperty name="user" property="*"/>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page import="SQLUtilities.Utilities" %>
<%@ page errorPage="myError.jsp?from=QueryFA.jsp"  %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>OlympiChronicles Query : FLAGS and ANTHEMS</title>

[automatically generated code...]

    <p>Please select the options for your query:</p>
    <form action="QueryResultFA.jsp" method="post" name="form1" >

      <select name="select"
onChange="MM_showHideLayers('Layer16','','show')">
        <option value="0" selected>Select Country</option>
        <%= Utilities.getCountries() %>
      </select>
<p>
        <input type="submit" name="Submit" value="Submit">
        <input type="reset"  value="Reset">
                        </p>
      </div>
      </form>
  </div>
</div>
  <p>   <br>
```

The code presented in this section is half of the code created for the web interface. For each of the query pages presented here there is a corresponding results page that shows the results of the query created by the user.

There have been updates that were made to the website and therefore the queries that had not been implemented, are now, therefore those pages look more or less similar to the ones presented here, what changes is the actual Java code to get the results.

7.2 Data Beans

SEQuery.java (Sport Event Query)

```
package SQLUtilities;

import java.sql.*;
import java.util.*;

public class SEData implements java.io.Serializable {

   String subsport;
   String eventt;
   String gender;
   private Connection db = null;

   public SEData()
   {
      subsport = "";
      eventt = "";
      gender = "";

      dbConnect ();
   }

   private void dbConnect ()
   {

      if (db == null)
      {
```

SEQuery.java (Sport Event Query Continued...)

```
public void setEventt ( String value )
{
   eventt = value;
}

public String getSubsport() { return subsport; }
public String getEventt() { return eventt; }


public String getCurrentRecord ()
   throws SQLException

{
   int index = eventt.indexOf (":") + 1;
   gender = eventt.substring (index).trim();
   String trimevent = eventt.substring (0,index - 1).trim ();

   boolean recordExists = false;

   String result = "";

   // Create a Statement
   Statement stmt = db.createStatement ();
```

SEQuery.java (Sport Event Query Continued...)

```
    public String getWorldRecord ()
     throws SQLException

  {
     int index = eventt.indexOf (":") + 1;
     gender = eventt.substring (index).trim();
     String trimevent = eventt.substring (0,index - 1).trim ();

     boolean recordExists = false;

     String result ="";

     // Create a Statement
     Statement stmt = db.createStatement ();

     // Select the ENAME column from the EMP table
     ResultSet rset = stmt.executeQuery ("select athlete, A.abrev, record, A.year
from currentWR A where subsport='" +subsport + "' and event ='" + trimevent
+ "' and gender = '" + gender + "'");
```

TMCData.java (Top Medal Countries Query)

```
package SQLUtilities;

import java.sql.*;
import java.util.*;

public class TMCData implements java.io.Serializable {

   String subsport;
   String eventt;
   String gender;
   private Connection db = null;

   public TMCData()
   {
      subsport = "";
      eventt = "";
```

TMCData.java (Top Medal Countries Query Continued...)

```
public String getSubsport() { return subsport; }
public String getEventt() { return eventt; }

public String getResults ()
   throws SQLException

{
   int index = eventt.indexOf (":") + 1;
   gender = eventt.substring (index).trim();
   String trimevent = eventt.substring (0,index - 1).trim ();
   int i = 1;
   boolean first = true;
```

TMCData.java (Top Medal Countries Query Continued...)

```
        result += "</table><br><br>";

        if (!recordExists)
        {
           result = "no data has been captured for your selections... please t\
ry again.";
        }
```

CPHData.java

```
package SQLUtilities;
import java.sql.*;

public class CPHData implements java.io.Serializable {

   String abrev;
   String country;
   private Connection db = null;
   String firstyear;
   String totalyears;

   public CPHData()
   {
      country = "";
      abrev = "";
      firstyear = "";
      totalyears = "";

      dbConnect ();
   }
private  void dbConnect ()
   {

      if (db == null)
      {
         try {
            DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
            db =
               DriverManager.getConnection
("jdbc:oracle:thin:@dbserv.dc.umd.edu:1521:NR424", "nr42418", "skidoo23");
         } catch (Exception ee){ System.out.println("Error: " + ee); }
      }
```

CPHData.java (Continued...)

```
   public  String getCountry()
   {
      return country;
   }
   public  String getAbrev ()
   {
      return abrev;
   }
   public  String getFirstyear ()
   {
      return firstyear;
   }
   public  String getTotalyears ()
   {
      return totalyears;
   }
      public  String getCountries ()
      {
         String result = "";

         try
         {
            Statement s = db.createStatement ();
            ResultSet rset = s.executeQuery ("select country from countryAbrev");

            while (rset.next ())
            {
               result += "<option value=\"" +rset.getString
(1)+"\">"+rset.getString (1)+"</option>";
            }
         }
         catch (Exception ee)
         {
            System.out.println ("Error populating UserData: " + ee.toString ());
            return "";
         }

         return result;
      }
      public  void setAbrev ()
      {
         String result = "";

         try
         {
```

CPHData.java (Continued...)

```
            while (rset.next ())
            {
               result = rset.getString (1);
            }
         }
         catch (Exception ee)
         {
            System.out.println ("Error populating CPHData: " + ee.toString ());
         }

         System.out.println ("setting abrev to: " + result);
         abrev = result;
      }
      public void setResult ()
      {
         setAbrev();

         try
         {
            Statement s = db.createStatement ();
            ResultSet rset = s.executeQuery ("select firstyear, totalyears from
countrypartici\
   pation where abrev ='" + abrev + "'");

            System.out.println (abrev);
            while (rset.next ())
            {
               firstyear = rset.getString (1);
               totalyears = rset.getString (2);
            }
         }
         catch (Exception ee)
         {
            System.out.println ("Error getting results CPHData: " + ee.toString ());
         }

      }
   }
```

UserData.java

```
package SQLUtilities;

import java.sql.*;

public class UserData implements java.io.Serializable {

    String years;
    String[] posterMedal;
    String country;
    String abrev;
    private Connection db = null;

    String countryyear;
    public UserData()
    {
       years = "";
       countryyear = "";
       abrev = "";
       posterMedal = new String[] {"1"};
       dbConnect ();
    }
    private void dbConnect ()
    {

       if (db == null)
       {
          try {
             DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
             db =
                DriverManager.getConnection
("jdbc:oracle:thin:@dbserv.dc.umd.edu:1521:NR424",
                                    "nr42418", "skidoo23");
          } catch (Exception ee){
             System.out.println("Error: " + ee);
          }

       }
    }

    public void setCountryyear ( String value )
```

UserData.java (Continued)

```
    public void setPosterMedal ( String[] values ) {  posterMedal = values;    }

    public void setSelect2 ( String value ) { country = value;   }

    public String getSelect() { return years; }

    public String getCountryyear() { return countryyear; }

    public String[] getPosterMedal() { return posterMedal; }

    public String getSelect2() { return country; }
    public String getAbrev ()
    {
       String result = "";
       try
       {
          Statement s = db.createStatement ();
          ResultSet rs = s.executeQuery ("select abrev from countryAbrev where
country like'%" + country + "%'");
          while (rs.next ())
          {
             result = (rs.getString (1)).trim ();
          }
       }
       catch (Exception ee)
       {
          System.out.println ("Error populating UserData: " + ee.toString ());
          return "";
       }
       abrev = result;
       return result;
    }

    public String isCbSelected(String s)
    {
       boolean found=false;
       if (!posterMedal[0].equals("1"))
```

UserData.java (Continued...)

```
    public String getGold ()
    {
       String result = "";
       try
       {
          Statement s = db.createStatement ();
          ResultSet rs = s.executeQuery ("select count(count (medal)) from
individualwins where country like '%" + abrev + "%' and year='" +
countryyear + "' and medal='gold' group by sport, subsport, event, gender");
          while (rs.next ())
          {
             result = rs.getString (1);
          }
       }
       catch (Exception ee)
       {
          System.out.println ("Error getting goldCount in UserData: " +
ee.toString ());
          return "";
       }

       return result;
  }
    public String getSilver ()
    {
       String result = "";

       try
       {
          Statement s = db.createStatement ();
```

UserData.java (Continued...)

```
    public String getBronze ()
    {
       String result = "";

       try
       {
          Statement s = db.createStatement ();
          ResultSet rs = s.executeQuery ("select count(count (medal)) from
individualwins where country like '%" + abrev + "%' and year='" +
countryyear + "' and medal='bronze' group by sport, subsport, event, gender");
          while (rs.next ())
          {
             result = rs.getString (1);
          }
       }
       catch (Exception ee)
       {
          System.out.println ("Error getting goldCount in UserData: " +
ee.toString ());
          return "";
       }
       return result;
    }
```

MPCData.java

```
package SQLUtilities;

import java.sql.*;

public class MPCData implements java.io.Serializable {

   String year;
   String YRyear;
   private Connection db = null;

   public MPCData()
   {
      year = "";

      dbConnect ();
   }

   private  void dbConnect ()
   {

      if (db == null)
      {
```

MPCData.java (Continued...)

```
    public String getNonPartCountriesCount ()
    {
       String result = "";
       System.out.println ("The year to search: " + year + " and the YR is: " +
  YRyear);
       try
       {
          Statement s = db.createStatement ();
          ResultSet rset = s.executeQuery ("select count (country) from
countryAbrev CA, countryYears CY where CY.abrev = CA.abrev and " +
YRyear + "='N'");

          while (rset.next ())
          {
             result += rset.getString(1);
          }
       }

       catch (Exception ee)
       {
```

MPCData.java (Continued...)

```
                else
                {
                   result += "<td><div align=\"left\">(" + rset.getString(2)+ ")
" + rset.getString(1) + "</div></td></tr>";
                }
                i++;
             }
             if (i % 2 == 0)
             {
                result += "<td><div
align=\"center\"> </div></td></tr>";
             }
          }
          catch (Exception ee)
          {
```

MPCData.java (Continued...)

```
            if (i % 2 == 0)
            {
               result += "<td><div
align=\"center\"> </div></td></tr>";
            }
         }
         catch (Exception ee)
         {
            System.out.println ("Error populating Non participating countries
MPCData: " + ee.toString ());
            return "";
```

MPCData.java (Continued...)

```
        try
        {
           Statement s = db.createStatement ();
           ResultSet rset = s.executeQuery ("SELECT Sub.country, Sub5.countryname,
GoldTotal,  SilverTotal, BronzeTotal, MedCount as Total FROM (SELECT country, count
(MedalCount) MedCount FROM (SELECT country, count (medal) MedalCount FROM
individualwins IW, olympicsite OS WHERE IW.year=OS.year and OS.year='" + year + "'
```

CEHData.java

```
package SQLUtilities;

import java.sql.*;
import java.util.*;

public class SEHData implements java.io.Serializable {

   String discipline;
   String event;
   private Connection db = null;

   String year;
   String subsport;
   String newevent;
   String gender;
   String medal;
   String country;
   String athlete;

   int currentRow;
   int rowCount;

   List yearList;
   List subsportList;
   List neweventList;
   List genderList;
   List medalList;
   List countryList;
   List athleteList;

   public SEHData()
   {
      discipline = "";
      event = "";
      dbConnect ();

      setYear ("");
      setSubsport ("");
      setNewevent ("");
      setGender ("");
      setMedal ("");
      setCountry ("");
      setAthlete ("");
      yearList = new ArrayList ();
```

CEHData.java (Continued...)

```
        genderList = new ArrayList ();
        medalList = new ArrayList ();
        countryList = new ArrayList ();
        athleteList = new ArrayList ();

        currentRow = 0;
        rowCount = 0;
    }

    private void dbConnect ()
    {

        if (db == null)
        {
            try
            {
                DriverManager.registerDriver
                    (new oracle.jdbc.driver.OracleDriver());

                db = DriverManager.getConnection
                    ("jdbc:oracle:thin:@dbserv.dc.umd.edu:1521:NR424",
                     "nr42418", "skidoo23");
            }
            catch (Exception ee)
            {
                System.out.println("Error: " + ee);
            }

        }
}

    public void setDiscipline ( String value )
    {
        discipline = value;
    }
    public void setEventt ( String values )
    {
        event = values;
    }
    public void setYear ( String values )
    {
        year = values;
    }
```

CEHData.java (Continued...)

```
    public void setNewevent ( String values )
    {
       newevent = values;
    }
    public void setGender ( String values )
    {
       gender = values;
    }
    public void setMedal ( String values )
    {
       medal = values;
    }
    public void setCountry ( String values )
    {
       country = values;
    }
    public void setAthlete ( String values )
    {
       athlete = values;
    }
    public String getDiscipline() { return discipline; }

    public String getEventt() { return event; }

    public String getYear() { return year; }
    public String getSubsport() { return subsport; }
    public String getNewevent() { return newevent; }
    public String getGender() { return gender; }
    public String getMedal() { return medal; }
    public String getCountry() { return country; }
    public String getAthlete() { return athlete; }
    public boolean populate ()
    {
       if (yearList.isEmpty ())
       {
          try
          {
             Statement s = db.createStatement ();
             ResultSet rs = s.executeQuery("select * from individualwins where
subsport ='" + discipline + "' and event ='" + event + "'");
```

CEHData.java (Continued...)

```
            countryList.clear ();
            athleteList.clear ();

            rowCount = 0;
            while (rs.next ())
            {
               yearList.add (rs.getString ("year"));
               subsportList.add (rs.getString ("subsport"));
               neweventList.add (rs.getString ("event"));
               genderList.add (rs.getString ("gender"));
               medalList.add (rs.getString ("medal"));
               countryList.add (rs.getString ("country"));
               athleteList.add (rs.getString ("athlete"));

               rowCount++;
            }

         }
         catch (Exception e)
         {
            System.out.println ("Error populating SEHData bean: " + e.toString ());
            return false;
         }
      }
      return true;
   }

   public void setStartRow (int _start)
   {
      if (_start < rowCount)
      {
         currentRow = _start;
      }

   }
   public int nextRow ()
   {
```

CEHData.java (Continued...)

```
        setMedal ((String)medalList.get(currentRow));
        setCountry ((String)countryList.get(currentRow));
        setAthlete ((String)athleteList.get(currentRow));

        currentRow++;

        return currentRow;
    }
    public int getCurrentRow ()
    {
        return currentRow;
    }
```

TMAData.java

```
package SQLUtilities;
import java.sql.*;
public class TMAData implements java.io.Serializable {
   Connection db = null;
   public TMAData()
   {
      dbConnect ();
   }
     private  void dbConnect ()
      {
         if (db == null)
         {
            try {
               DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
```

TMAData.java (Continued...)

```
        while (rset.next ())
                {
                   result += rset.getString(1);
                }
           }
        catch (Exception ee)
           {
              System.out.println ("Error populating Non participating countries
MPCData: " + ee.toString ());
              return "";
           }

        return result;
     }
  }
```

Non-Bean File → Utilities.java

```
package SQLUtilities;

import java.sql.*;
import java.io.*;
import java.util.ArrayList;

public class Utilities
{
   private static Connection dbConnect ()
```

Non-Bean File → Utilities.java

```
        // Create a Statement
        Statement stmt = conn.createStatement ();

        // Select the ENAME column from the EMP table
        ResultSet rset = stmt.executeQuery ("select year from olympicsite");
        // Iterate through the result and print the employee names
        while (rset.next ())
           {
              //System.out.println (rset.getString (1));
              result += "<option value=\""+rset.getString (1)+"\">"+rset.getString
(1)+"</option>";
           }


        conn.close ();
        return result;
     }
     public static String getCountries ()
        throws SQLException
     {
        String result = "";
        Connection conn = dbConnect ();
```

Non-Bean File → Utilities.java

```
        // Create a Statement
        Statement stmt = conn.createStatement ();

        // Select the ENAME column from the EMP table
        ResultSet rset = stmt.executeQuery ("select country from countryabrev");
        // Select the ENAME column from the EMP table
        ResultSet rset = stmt.executeQuery ("select country from countryabrev");

        // Iterate through the result and print the employee names
        while (rset.next ())
           {
              //System.out.println (rset.getString (1));
              result += "<option value=\"" + rset.getString (1) + "\">" + rset.getString
(1) + "</option>";
           }


        conn.close ();
        return result;
```

Non-Bean File → Utilities.java

```
public static String outputPosterURL (String yearIn)
   throws SQLException

{
   String result ="";
   Connection conn = dbConnect ();

   // Create a Statement
   Statement stmt = conn.createStatement ();

   // Select the ENAME column from the EMP table
   String argue = "select poster from olympicsite where year='"+yearIn+"'";
   ResultSet rset = stmt.executeQuery (argue);

   result += "<tr><td><div align=\"center\"><img src=\"rset.getString
```

Non-Bean File → Utilities.java

```
        ResultSet rset2 = stmt.executeQuery ("select unique event from
individualwins where subsport='" + String.valueOf (subsport.get(j)) + "'");

        while (rset2.next ())
        {
           events += "\""+ rset2.getString(1) + "\",";
        }
        events = events.substring (0,events.lastIndexOf (","));

        result += "displine[\"" + String.valueOf (subsport.get(j)) +
           "\"] = [" + events + "];\n";
```

```
while (rset2.next ())
 {
    events += rset2.getString(1) + "|";
 }
 events = events.substring (0,events.lastIndexOf ("|"));

 result += "\"" + String.valueOf (subsport.get(i)) +
```

Non-Bean File → Utilities.java

```
            while (rset2.next ())
               {
                  years += rset2.getString(1) + "|";
               }

            years = years.substring (0,years.lastIndexOf ("|"));
            result += "\"" + String.valueOf (country.get(j)) +"*" + years + "\",\n";
            years = "";

          }

        result = result.substring (0,result.lastIndexOf (","));

        conn.close ();

        return result;
     }
     public static int getNumDisplines()
        throws SQLException

     {
        int result = 0;
        Connection conn = dbConnect ();

        // Create a Statement
        Statement stmt = conn.createStatement ();

        // Select the ENAME column from the EMP table
        ResultSet rset = stmt.executeQuery ("select count (unique subsport) from
individualwins");
       while (rset.next())
        {
           result = rset.getInt (1);
        }

        return result;
     }
     public static String getArcheryEvents ()
        throws SQLException
     {
        String result = "";
        Connection conn = dbConnect ();

        // Create a Statement
```

Non-Bean File → Utilities.java

```
        // Select the ENAME column from the EMP table
        ResultSet rset = stmt.executeQuery ("select unique event from individualwins
where subsport='Archery'");

        while (rset.next ())
        {
           //System.out.println (rset.getString (1));
           result += "<option>" + rset.getString (1) +
              "</option>";
        }

        conn.close ();
        return result;
     }
```

7.3 SQL Queries

All the SQL queries created are embedded in the Java code presented above.

8 USER MANUAL

8.1 How to Access OlympiChronicles

To access OlympiChronicles you need access to a computer and to the internet. The webpage works better if viewed with Internet Explorer 4.0 or greater. The URL for OlympiChronicles is http://dc.umd.edu:8081/index2.html

8.2 How to Navigate Through OlympiChronicles

Once at the OlympiChronicles website click on the torch in the middle of the introduction page to enter. This takes you to the welcome page where you will be redirected to the QuerySelect page where you will be able to select your query. There is a navigation bar on the left side of the screen with all the possible queries that you can make. To see what each query does, mouse over its name and a description will

appear. particular

If you are interested in that query, click on it and you will be redirected to that query. This is where you can choose your options for your query.

8.2.1 Sport Event Query

The Sport Event Query allows you to choose a discipline (sport or sub sport) and then an event from that discipline. The results are the Olympic record and the world record for that event, it shows you the name of the athlete who holds that record, the country, the result and the games in which the record was broken.

8.2.2 Sport Event History Query

The Sport Event History Query allows you to choose a discipline (sport or sub sport) and a particular event and it returns the history of that event. The results display the Olympic year, the event you chose, the gender (whether it was male or female), and the athletes' names and countries.

8.2.3 Country Participation Query

The Country Participation Query allows you to choose a country and the results are the country abbreviation, the first year the country participated in the summer Olympics, and the total number of years that the specific country participated in the summer Olympics.

8.2.4 Medal Count Query

The Medal Count Query allows you to choose a country (only countries that have won at least one medal are available to choose from) and an Olympic year and the results are the country, the total number of gold medals, total number of silver medals, total number of bronze medals and the total number medals won by that country on that specific Olympic year.

8.2.5 Medals per Country Query

The Medals per Country Query allows you to choose and Olympic year and the results are the list of countries that participated in the Olympics that year with their total number of gold medals, silver medals, bronze medals and total number of medals for that year. The results also contain a list of countries that participated that year but did not win any medals as well as a list of countries that did not participate in the summer Olympics of that year.

8.2.6 Top Medal Athletes Query

The Top Medal Athletes Query does not ask you to choose from any option, the query displays a list of athletes that have won at least three medals during the entire history of the summer Olympic Games.

8.2.7 Top Medal Countries Query

The Top Medal Countries Query allows you to choose a discipline (sport or sub sport) and an event from that discipline and the results are the country and the total number of medals won for that country for that particular event. This result represents the medal wins per country through the history of the Olympic games.

8.2.8 Year Record Broken Query

The Year Record Broken Query allows you to choose a discipline (sport or sub sport) and then an event from that discipline. The results are the progression of Olympic and world records broken for that discipline / event.

8.2.9 Posters and Medals Query

The Poster and Medals Query allows you to choose an Olympic year and the choose any of the following three options: poster, medal-front, and medal-back. Depending on what images you want to see, the result will be an image of the official poster and medal (front and back) for that year.

8.2.10 Flags and Anthems Query

The Flags and Anthems Query allows you to choose a country and an image of its flag will be displayed along with a link to an audio file with the country's anthem.

9 TESTING EFFORTS

9.1 Web Interface

The issue that needed attention in the web interface was input validation. For example, users cannot proceed to the results page if they do not make choice. This is achieved by hiding the "Submit" option until they have a made a choice. When they have to choose from various different options, as in the case of medal and poster, there is a function that checks what needs to be displayed to the user. All the queries were tested with various different inputs to make sure that they work and in the case that no results are found for that query for the particular options chosen by the user, then a

message is displayed letting the user know that.

9.2 Data Beans

The data beans are used to retrieve the information from the database once the user has made a selection or has created a query. The data validation is done in the client side therefore that is not necessary here. There are proper try and catch block specially to make sure that the connection to the database has not failed. To our knowledge, the beans work correctly as long as they are called in the right place from the JSP pages. Due to lack of time no further testing was done.

9.3 SQL Queries

The SQL queries were created using specific examples (i.e. when the input is a year, then using a specific year, etc) and they were tested with different inputs to assure the correctness of the results. Most of it was done by hand, matching results against the Olympics.org database results since that was the greatest source for the OlympicsDB. Most of the queries work correctly. A couple of queries do not return a complete set of results, but the results that generates are correct..

10 SYSTEM LIMITATIONS

The major limitation in developing this project was the lack of time and knowledge about the different technologies that were needed for the completion of the project. Event with this, Craig Shapiro managed to finish the website and the results for all the queries. Another very important factor was the lack of some data that was needed such as a complete set of results for the summer Olympic Games, records broken and a complete list of athletes that participated but did not win medals. Part of this information currently resides in the database but it is not complete.

11 POSSIBILITIES FOR IMPROVEMENTS

- Integrating or adding the missing data to the database (complete set of results and records)
- Optimizing the SQL queries (a couple of them take a few seconds to generate).

12 CREDITS

Craig Shapiro:

1. Research: ETL Tools, Tomcat (Servers in general), JSP, JavaBeans, SQL, Web Development, Summer Olympic Data, etc.
2. ETL: Kapow RoboSuite 5.5 learn and program to extract and load data into

database.

3. Database: all the tables were created by Craig Shapiro (most of the data was obtain through Kapow RoboSuite 5.5)
4. Web Interface: Design and implementation
5. Documentation: Phase I, Phase II + some diagrams, revisions and paper copies.
6. Java Code: All the files (Java and JavaBeans) were created by Craig Shapiro

Steffanie Orellana:

1. Research: ETL Tools, Tomcat, JSP, JavaBeans, SQL, Summer Olympic Data, etc.
2. Database: formatted data for Craig to create a couple of tables.
3. Web Interface: Design
4. Documentation: Phase I, Phase II, Phase III + some diagrams, revisions
5. SQL queries (with revisions and changes made by Craig Shapiro)

13 WEB SITE RESOURCES

Our most extensive source of data was the official Olympics website (Olympics.org and Olympic.it)

http://www.factmonster.com/ipka/A0114094.html
http://www.ex.ac.uk/trol/databank/olympics/index.htm
http://www.olympic.org/uk/index_uk.asp
http://www.e-magine.education.tas.gov.au/pl/olympics/index.htm
http://www.infoplease.com/ipsa/A0114094.html
http://www98.pair.com/msmonaco/Almanac/
http://www.athletics-heroes.net/athletics-heroes/stats_athletics/olympics/olympics.htm
http://www.runnersweb.com/running/rw_news_frameset.html?http://www.runnersweb.com/running/olympics.html
http://www.recordholders.org/en/links.html
http://www.hickoksports.com/history/olympix.shtml
http://www.olympic.it/english/game
http://www.flags-and-anthems.com/