A DBMS Engine Simulator to Assist in Classroom Understanding of Oracle Query Execution

Brett Allenstein, with Dr. Paul J. Wagner (faculty mentor)
Department of Computer Science
University of Wisconsin – Eau Claire
130 Phillips Science Hall
Eau Claire, WI 54702-4004
allensbs@uwec.edu and wagnerpj@uwec.edu

Abstract

Oracle's DBMS query execution process consists of a complicated series of steps which can be difficult for many students and professionals to understand. It is also difficult to find resources which provide a compiled description of this process and all of the components involved. We have researched Oracle's DBMS query execution process in order to compile the information available on this process. We have developed a simulation application that provides a simple interface to relay this compiled information to others in order to aid in the understanding of this process. We have designed our simulation application using several design patterns to allow for anticipated future changes resulting from additional research. We are enthused that our research and this simulation application will positively contribute to the knowledge base of Database Systems.

1 – Introduction

The query execution process of any database management system is a difficult concept for many students to grasp. A general understanding of how query execution is handled can be very helpful to students studying Database Systems concepts, and can ultimately help them to write more efficient queries. Although other institutions may take a different approach, the Database Systems course offered through the Computer Science department here at the University of Wisconsin – Eau Claire (hereafter UW – Eau Claire) focuses its time on studying how the Oracle database management system (DBMS) engine [3] handles this process.

The execution of a query in Oracle consists of a fairly complicated series of steps involving different areas of memory and disk. These components interact in specific ways to produce the result of a query passed to Oracle's DBMS. The complexity of this process makes it difficult to diagram and explain in the classroom by repeatedly drawing and erasing figures on the board. This teaching technique is time consuming, can be difficult for an instructor to perform, and can also be difficult for students to follow.

We believe that an interactive simulation is the most effective teaching tool for the instructor and learning aid for students. Visual simulations of similar such processes have previously been shown helpful to student understanding, so we think that such a simulation for Oracle's query execution process will have this same positive result.

2 – Background Investigation

Our initial work consisted of pulling together information on Oracle's query execution process from various resources. It was difficult to find a broad base of information on this process, as there were only a few resources to draw from. [1] contained the most structured version of the query execution process, including a well defined outline of the steps involved. We expanded on this initial information as we continued researching the process in other references. These other sources of information included manuals for later versions of Oracle, [2], which contained more specific information on the internal components involved in the query execution process, but not a lot of information on the process itself. Some of the information we used also came from what Dr. Wagner had compiled for the Database Systems course offered at UW – Eau Claire. There were even less online resources relating to our research, which resulted in most of our work being based on the material from these two Oracle manuals.

We also spent time looking for other research done in this same area of Database Systems, as we were interested in finding others that might have developed a similar simulation tool or compiled information on Oracle's query execution process. We found no such tool and very little other research done in this area. The apparent lack of research in this area is what led us to develop our current system, and we hope our work can

provide a positive contribution to the knowledge base of Database Systems and the affectivity of how it is taught in the classroom.

3 – Process / Work Done

3.1 – Design of the View

The initial focus of our research was into how we could effectively display Oracle's query execution process in a simulation. The application needed to provide an interface to the student and instructor which could contribute to the classroom learning process. The application also needed to be extensible, as we were constantly expanding on what we knew about the process and the functionality that we wanted to provide in the application.

3.1.1 – View: Key Concepts

Our research had produced three main views of the DBMS components involved in the query execution process. These components reside in areas of main memory and disk, so these were two views we wanted to display to the user. The DBMS components in main memory and on disk interact with each other during the process, so we also found it important to provide an overview of both of these areas.

The DBMS components that are involved in Oracle's query execution process participate in different ways at different points in time during the progression of the process. The activity of each component needed to be displayed to the user somehow as the query execution process progressed. Each DBMS component also needed to be discretely defined to avoid any confusion as to what each component was and what it did.

Finally, we needed the view to provide a way for the user to step through the query execution process. This simulation must allow the user to step forward and backward in the process, and additionally provide specific information about what was happening at each step in the process.

3.1.2 – Development of the Prototype

Our prototype consists of three main panels which we have found to address all of the key concepts mentioned above. These three panels are labeled in *Figure 1*, as the *tabbed view panel*, *control panel*, and *definition panel*. In addition to these three panels, our prototype also provides a place to display the current query being executed, which, at this point, is a single static query (more on this later).

The *tabbed viewing panel* allows the user to switch between a view of main memory, a view of disk, and the overview of these two areas. Each tab panel displays the DBMS components found in that area, and each component is drawn on the screen with a color indicating the component's current state of activity. A color key is displayed in the corner of the tabbed viewing pane so that the user can determine the state of activity of each component in that view. The *tabbed viewing panel* can be found in *Figure 1* towards the top of the graphical user interface.

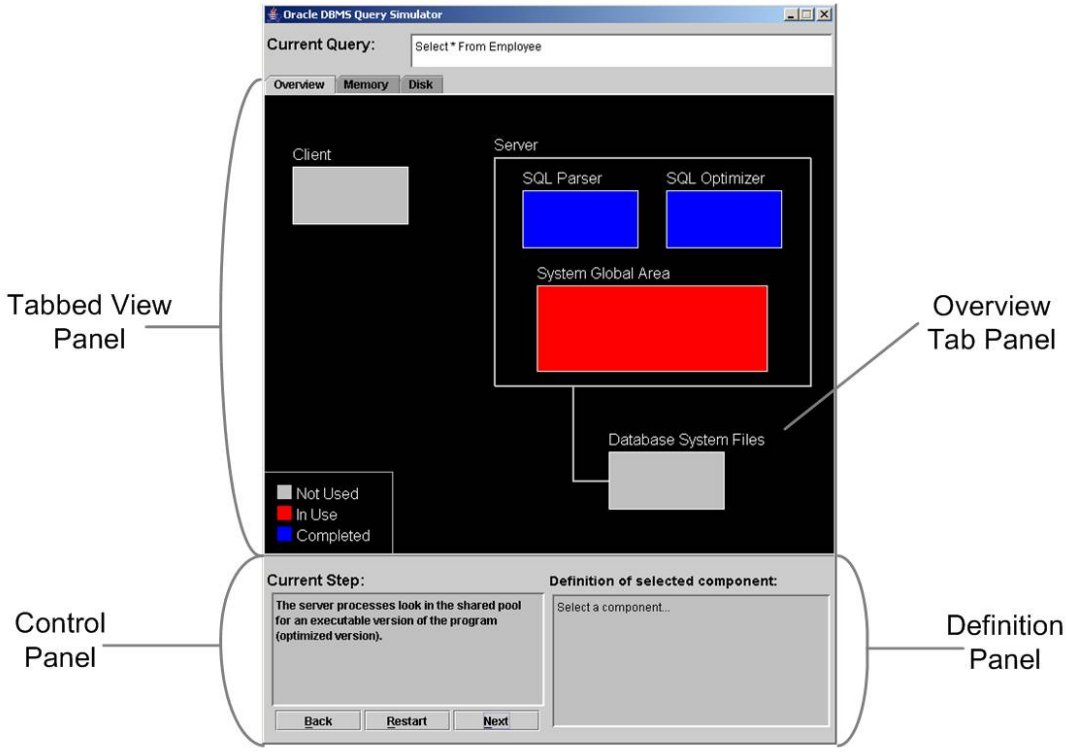

Figure 1: A view of our application's entire GUI, containing the three main panels: *tabbed view panel*, *control panel*, and *definition panel* along with the *overview tab panel*

Figure 1 also includes the *overview tab panel* which displays the interaction between the client, areas of main memory, and areas of disk during the processing of a query. This view of the query execution process is included to specifically show this interaction between these areas. A component representing the client requesting a query to be processed by the Oracle DBMS is displayed on this tab panel along with a component representing all of the database files found on disk (Database System Files). The Server where the database instance is running is also displayed on this panel, and it contains the SQL Parser, SQL Optimizer, and the System Global Area, where most of Oracle's main memory structures exist.

The next tab provided in the *tabbed viewing panel* is the *memory tab panel*. This panel displays the main memory constructs contained in, and some processing done by, the Oracle DBMS instance. The System Global Area is outlined in this view so we can display the various internal components that it contains. These components are involved

in caching data and other information needed in the processing of a query, and they include the Data Cache, Shared SQL Pool, Redo Log Buffer, and Data Dictionary Cache. This view also displays the SQL Parser and Optimizer which are involved early in the processing of an SQL query. This view is displayed in *Figure 2*, and it is a more detailed view of the Server found in the *overview tab panel*.

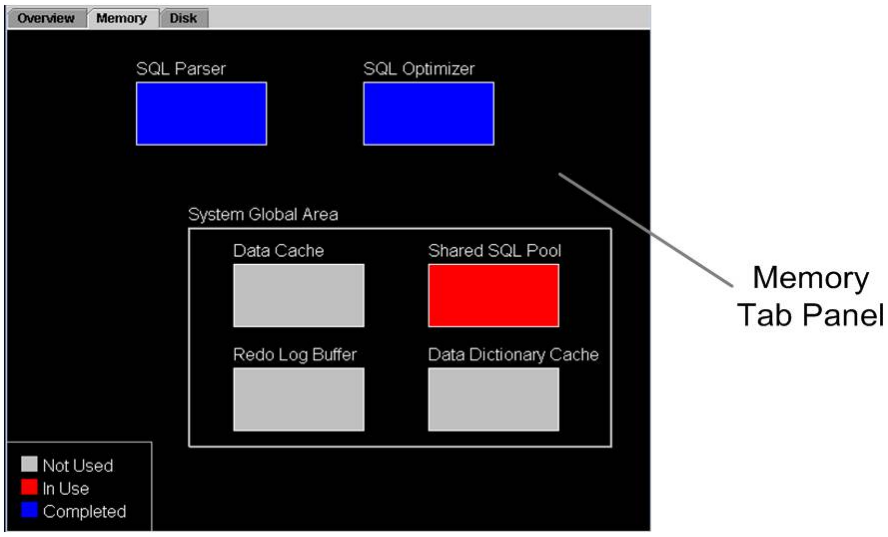

Figure 2: *Memory tab panel* displaying the main memory constructs used in Oracle's DBMS query execution

The final view of the components involved in the processing of a query by Oracle's DBMS engine is the *disk tab panel*. This panel displays the actual datafiles, redo log files, and rollback segments that are used and/or cached by the DBMS engine and then stored on disk. These components are involved in data storage and data backup strategies for a database.

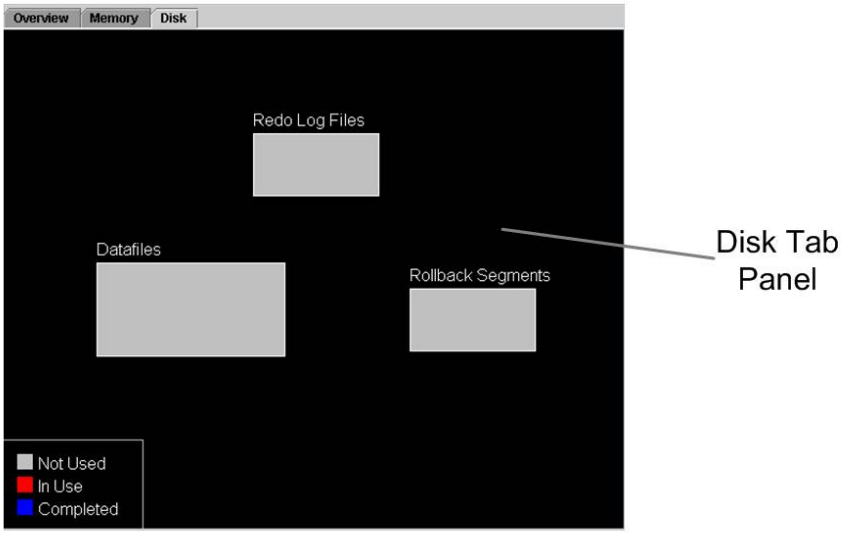

Figure 3: *Disk tab Panel* which displays the components on disk which are used in Oracle's DBMS query execution

There is also a *control panel* provided on the GUI which provides information on the current step of execution in Oracle's query execution process. This panel displays a textual description of what is occurring in the current step of the simulation. It also gives the ability to move to the next step, previous step, or first step in the query execution process.

Finally, the prototype provides a *definition panel* that gives the user access to the formal definition of each of the components currently being displayed to the user. The user can select a DBMS component in the current view by clicking on it, and its definition will be displayed on this panel. The definition is cleared when the view is changed, so there is no confusion as to which components are being displayed and the one that is defined on this panel.

3.1.3 – View Implementation / Software Design

The implementation of the view of our prototype involved some careful design decisions. We believe our design provides a balanced separation between the view and the domain. We also believe that our design will allow for future expansion when more information about the process is brought to light or when small modifications to the process occur.

Each of the panels placed on the tabbed viewing pane is a child class of a class we developed call DBMSViewPanel. The DBMSViewPanel class provides the basic functionality that every viewing panel needs. This includes holding and drawing a collection of DBMS components for that view, and drawing the color key which must be displayed in every view. The DBMSViewPanel class also provides a mechanism to determine which component was clicked in that view when a DBMS component definition is requested. The child classes of DBMSViewPanel determine their own layout by specifying which components they wish to display, and where they wish to display them. A child class of DBMSViewPanel can also provide some custom drawing, which allows the "Server" and "System Global Area" outlines to be drawn on the overview and memory panels respectively.

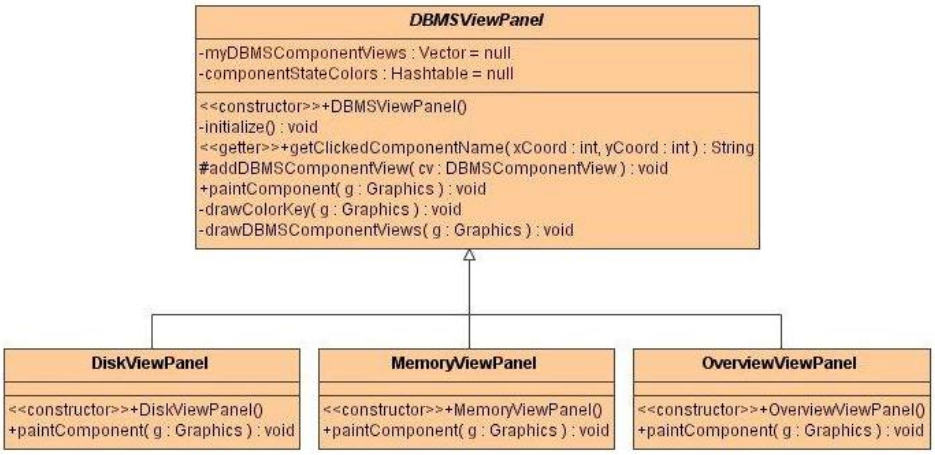

Figure 4: Class diagram for tabbed viewing panels

This design of the viewing panels makes it much easier to modify any view or even add a new view. Components can be added, removed, or rearranged in any of the child classes of DBMSViewPanel without worrying about how they will be drawn or how to determine which component was selected by a request for a component definition. Additional views of Oracle's DBMS can also be developed, because a new child class can be created and the components that will be displayed in this new view can simply be added and arranged appropriately. Any custom drawing for this new view can also be accommodated by this design. This is a very useful design, because there are "black boxes" in Oracle's query execution process that could be elaborated on, like the system global area is treated between the overview and memory views. Additional views of this process are a very real possibility for future expansion.

There is also a singleton class called ComponentStateManager that is used by the view to determine which color to draw each component with. This decision on the drawing color is made within the DBMSViewPanel class, since it is responsible for drawing each of the components. The ComponentStateManager is one of the few interfaces that is used between the view and the domain, and is one of the constructs used to create a good balance in the interaction between the domain and the view.

The DBMS component definition lookup mechanism is handled by the ComponentDefinitionManager class, which is also a singleton. This class uses an XML file which contains all of the DBMS component definitions. The Component-DefinitionManager parses the XML file to extract the definition for each component used within the application on the first request for a definition. This class will then hold onto the definition information for the duration of the simulation, and return the definition for each component that the user selects. This design allows for the easy manipulation of the text used to define each component, so that an instructor can provide their own modified definitions. It also allows for other minor changes in the component definitions that we have developed which could result from further research.

3.2 – Design of the Domain

Initial expectations are rarely the final expectations for any project, and this was exactly the case as we progressed through researching Oracle's query execution process. We have designed a system that simulates a static query with static process and data behind it. Any decisions that need to be made in this process are posed to the user (e.g. Is the executable version of the query in the shared pool?). As ideas for potential expansion were developed, the design of the domain of our prototype became more important.

3.2.1 – Domain: Key Concepts

Database Management System components are areas of main memory or disk that play key roles in the processing of a query. These components can contain data or other information that can be used in the query execution process, but our static simulation

focuses on the state of activity of each of these DBMS components. At any point in time, a component might not have been used yet, might be in use, or might have already completed its job. This is the type of information that we keep track of for each component in our simulation.

Oracle's query execution process consists of a sequence of steps. Each of these steps marks a point in the process where certain DBMS components are performing certain tasks. It is important to know this information as the simulation progresses, so that it can be displayed to the user. The order in which certain steps occur can depend on what data is contained in certain DBMS components. An example of this occurs when a decision needs to be made as to whether data should be retrieved from disk. The retrieval of data from disk can be skipped if that data is already in the data cache.

3.1.2 – Development of Solutions to Modeling the Domain

Since Oracle's query execution process consists of a fairly complicated series of steps, we decided to develop classes that represented each step in the process. These classes contained the textual information about what was happening at that point in the query execution process. These objects also allowed us to give meaning to the abstract concept of a step or a point in time during the query execution process. This way our simulation could hold onto the point in time or step that was currently being executed.

Our query execution simulation also keeps track of the state of activity for each DBMS component that is displayed to the user. The activity of each component is specific to the step that is currently being executed during the processing of the query. Our initial design tied the information about the state of each DBMS component to each query step, but this did not seem like the appropriate place for such information. We later pulled this information out into a single class that would manage the state of activity for each DBMS component. This approach seemed much more appropriate, because now the query execution steps could interact with this one object to specify the state of activity for each DBMS component.

In addition to representing the steps that occur in Oracle's query execution process, we also needed a way to transition between these steps. This responsibility was initially assigned to each query step object as described above, but this approach required that each query step know its relationship to other query steps. This wasn't the appropriate place to put this responsibility, so we modified our design by pulling this decision making process out into another class. The logic used for transitioning between steps was now in a single place in our code, and not distributed among various query steps. This decision was also made to allow for possible future expansion of the application which will be discussed in later sections.

3.2.3 – Implementation / Software Design

Each step in the query execution process is modeled as a child class of the abstract QuerySimulationStep class. This parent class provides a place to keep a step number, name, and textual description for each query execution step, and requires that each class override a method to set the state of activity for each DBMS component in the simulation. Any child class of the QuerySimulationStep class provides its own definition for the instance variables of this parent class. This design provides a base class which defines the information and interface that each concrete query execution step should provide to the simulation application. The design structure for a QuerySimulationStep also allows for the addition of new steps that may be determined through future research.

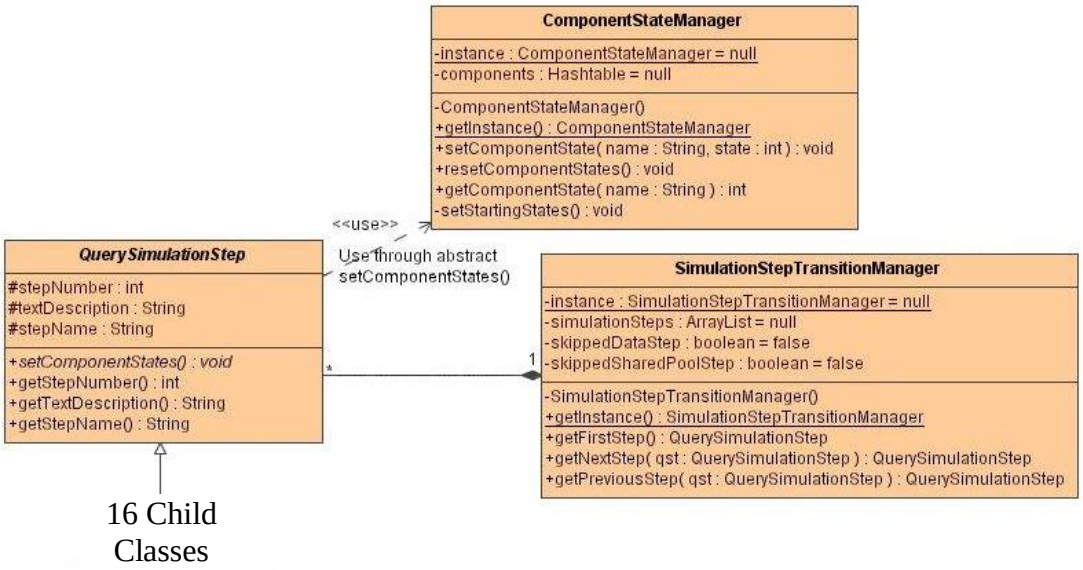

Figure 5: Class diagram for main domain classes

As was mentioned above, our simulation application utilizes a singleton class called the ComponentStateManager that keeps track of the state of activity for each DBMS component in the simulation. This means any DBMS component in the simulation will be specified as "not used," "in use," or "completed" at any point in time during the simulation. This class is used in the domain when a QuerySimulationStep is used to set the activity of each DBMS component. The view uses the ComponentStateManager to determine which color is used to draw a DBMS component on the screen. The ComponentStateManager is essentially a hash map of component names to their corresponding activity, and this object could be expanded upon to map each component name to an actual object that would represent the DBMS component's state of activity along with other additional information useful to the simulation (this is useful for the interaction with an actual Oracle DBMS, explained in Future Work).

Finally, we designed a class to handle the transition from one QuerySimulationStep to another called SimulationStepTransitionManager. The SimulationStepTransition-Manager simply orders and holds onto a collection of QuerySimulationStep instances, and then returns the appropriate object when the next step, previous step, or first step is

requested. This class is also a singleton object, since only one such step manager would be needed in our application. This class constructs each concrete step and assigns it a step number, which is used to determine the transitions between steps.

As was stated above, the responsibility for transitioning between steps was initially assigned to the QuerySimulationStep itself, but it was important to place this logic in one place. This design along with the QuerySimulationStep structure makes it much easier to incorporate new steps in this process that might be determined at a later date. This class was also important because it provides one extra level of abstraction which might help in making a smooth transition between a statically executed simulation to a dynamically executed simulation that could interact with an actual Oracle DBMS instance (discussed in Future Work).

4 – Future Work

Our research into this area of Database Systems has sparked numerous ideas in the area of future work. We have designed the simulation application itself so that it might be extended in numerous ways and areas. There is definitely future research needed in the area of Oracle DBMS query execution and plenty of work left to be done on our prototype simulation application.

We would like to do quite a bit more research into more specific details of how Oracle handles query execution. We suspect that there are minor details in our understanding of the process that we need to iron out through more research. This was much more apparent as we finished the development of our prototype simulation application, because we were continuously questioning more and more specific parts of the process, and trying to determine the best way to represent them. The design of our prototype should allow for these types of minor adjustments quite easily.

There are numerous additional features that we would like to add to our prototype application. We would specifically like to add a view of the execution plan that Oracle develops during the query execution process to show how some optimizations are made to certain queries. We would also like to add some additional configuration mechanisms that would allow the user to more easily modify the text displayed to explain a query execution step and the component states for each of the query execution steps. Currently this must be done within our code, and then the code must then be recompiled. These types of additional features would be helpful to instructors who wish to configure the application according to their own preferences, and it may also be helpful to change the application if new information about the process is brought to light.

Our largest and most interesting extension on this query simulator is the incorporation of a mechanism that could dynamically interact with an actual Oracle DBMS instance. This was an idea that we had in the back of our minds from the beginning of our research. We planned to develop a static model of the query execution process first, but as we did this, we were careful to try to design the static system so that it might “scale well” for the

extension to a dynamically executed system that could interact with an Oracle DBMS instance.

5 – Conclusion/Evaluation

We have plans to use this application in the Database Systems course offered by the University of Wisconsin – Eau Claire Computer Science Department. Unfortunately, the results of the formal testing of our simulation application in this course will not be available to us by the time this paper is submitted. We have confidence in the application and anticipate positive results from this formal testing, but we do need to disseminate the results of our research and do more of these types of formal evaluation of our work.

One of the positive sides to the apparent lack of material on Oracle's query execution process is that our research will be a good candidate for making a contribution to Database Systems. The information that we were able to gather and compile should be very useful to instructors attempting to convey these concepts to their own students and to professionals that may be trying to learn about them on their own.

We are confident that we have produced a very useful application for teaching this material to others and for the future expansion of the application for the numerous reasons discussed above. Our prototype application brings together the information available to us about Oracle's DBMS query execution process, and it has been designed to allow for changes to the domain processes, presentation, and configuration. We hope that this design will prove itself to be useful when future expansions are actually implemented.

References

[1] Abbey, Correy and Abramson, "Oracle 8i: A Beginner's Guide", Osborne/Oracle Press, 1999.

[2] Loney and Koch, "Oracle 9i: The Complete Reference", Osborne/McGraw Hill, 2002.

[3] Oracle DBMS, http://www.oracle.com