RULE BASED GRADE CALCULATION AND REPORTING USING DATAMART

Vikram Krishan
B.S, Thapar University, India, 2000

PROJECT

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

SPRING
2010
RULE BASED GRADE CALCULATION AND REPORTING USING DATAMART

A Project

by

Vikram Krishan

Approved by:

__________________________________, Committee Chair
Meiliu Lu, Ph.D.

__________________________________, Second Reader
Xin Ren, Ph.D.

__________________________________
Date
Student: Vikram Krishan

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

__________________________, Graduate Coordinator _____________________
Cui Zhang, Ph.D.                          Date

Department of Computer Science
Abstract

of

RULE BASED GRADE CALCULATION AND REPORTING USING DATAMART

by

Vikram Krishan

Student grade calculation among different subjects is a key part in determining the academic performance of student. Teacher evaluates the student performance grade in a subject on the basis of different elements for example quiz, homework assignments, midterm and final examinations, class projects etc. The rules for the calculation of final subject grade on the basis of different elements are defined by teacher which varies among different subjects. This project automates the student grade calculation process and provides enough flexibility faculty members to run analysis on the class performance.

In the initial analysis for the project I have found that there are many limitations of the grading process used by teaching staff among university departments. University departments have very diverse rules in calculating the subject grades which results in complex manual calculations by the teacher to determine the final subject grade. The current tools such as WebCt has some technical limitations. The current process does not give enough alternative options to the teacher to adjust the Grade calculation rules. Effective reporting is missing which can help teacher to understand the effectiveness of
improve the grade calculation process and make it more automated and visible by providing effective historical analysis for the teacher.

______________________, Committee Chair
Meiliu Lu, Ph.D.

______________________
Date
ACKNOWLEDGMENTS

I wish to express my sincere thanks and gratitude to my advisor and distinguished faculty member of Computer Science Department Professor Meiliu Lu for guiding me through this project. I value and thank her for the help, motivation and support without which this work would never have reached completion. I am also thankful to Professor Xin Ren, Division of Criminal Justice for her valuable input going towards this project.
TABLE OF CONTENTS

Page

Acknowledgements ................................................................................................................... vi

List of Tables .......................................................................................................................... x

List of Figures ......................................................................................................................... xi

Chapter

1. INTRODUCTION .............................................................................................................. 1
   1.1 Objective ................................................................................................................. 1
   1.2 Purpose of this Project ......................................................................................... 2
   1.3 Overview .............................................................................................................. 3
   1.4 Project Methodology ......................................................................................... 4

2. USER REQUIREMENTS ASSESSMENT ....................................................................... 6
   2.1 Overview .............................................................................................................. 6
   2.2 User Requirements Process ........................................................................... 7
   2.3 Requirements Areas ......................................................................................... 8
   2.4 Requirements Questions/Answers .................................................................. 9
   2.5 Use Cases ......................................................................................................... 12
   2.6 Functional Specifications .............................................................................. 15

3. PROJECT PLAN .............................................................................................................. 18
   3.1 Project Tasks Identification and Deliverables .............................................. 18
   3.2 Project Tasks Schedule Plan ......................................................................... 19
4. DATAMART FOR GRADING SYSTEM .......................................................... 21
   4.1 Datamart ............................................................................................... 21
   4.2 Relational Database Design vs Datamart Design .................................. 22
   4.3 Dimensional Data Modeling Key Concepts ......................................... 24
   4.4 Dimensional Modeling ........................................................................ 25
   4.5 Dimension Attributes ......................................................................... 26
   4.6 Dimension Tables ................................................................................ 26
   4.7 Fact Table ........................................................................................... 29
   4.8 Star Schema ....................................................................................... 31
   4.9 Snowflake Schema ............................................................................. 32

5. SYSTEM DESIGN ...................................................................................... 33
   5.1 System Architecture ........................................................................... 33
      5.1.1 Grade Rules Storage Module ....................................................... 33
      5.1.2 Data Capture Module ................................................................. 34
      5.1.3 Grade Calculation Module .......................................................... 35
      5.1.4 Data Reporting Module ............................................................... 35
   5.2 Database Modeling .............................................................................. 35
      5.2.1 Conceptual Model ...................................................................... 36
      5.2.2 Logical Model ............................................................................. 36
      5.2.3 Physical Data Model .................................................................... 38
      5.2.4 Datamart Star Schema ................................................................. 40
   5.3 Front End Wireframes ......................................................................... 41
5.3.1 Subject Interface ................................................................. 41
5.3.2 Class Interface ..................................................................... 44
5.3.3 Faculty Interface ................................................................. 45
5.3.4 Student Interface ................................................................. 46
5.3.5 Reporting Interface ............................................................. 47

6. ETL ............................................................................................ 49
   6.1 Loading the Data into Tables ................................................. 49
   6.2 Student Dimension ............................................................. 49
   6.3 Course Dimension ............................................................. 49
   6.4 Grades Rules Dimension .................................................... 50
   6.5 Grading Elements Dimension ............................................ 50
   6.6 Time Dimension ............................................................... 50

7. TESTING ...................................................................................... 51
   7.1 Test Cases ............................................................................ 51
   7.2 Sample Run ......................................................................... 52

8. CONCLUSION ............................................................................. 58

Appendix A MySQL ........................................................................ 60
Appendix B Project Code .............................................................. 63
Bibliography .................................................................................. 96
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Table 1.1 Project Development Lifecycle</td>
<td>5</td>
</tr>
<tr>
<td>2.</td>
<td>Table 2.1 Key Requirement Areas</td>
<td>8</td>
</tr>
<tr>
<td>3.</td>
<td>Table 3.1 Project Schedule</td>
<td>19</td>
</tr>
<tr>
<td>4.</td>
<td>Table 4.1 OLTP vs OLAP Database</td>
<td>23</td>
</tr>
<tr>
<td>5.</td>
<td>Table 4.2 Relational Data</td>
<td>27</td>
</tr>
<tr>
<td>6.</td>
<td>Table 4.3 Multidimensional Data</td>
<td>28</td>
</tr>
<tr>
<td>7.</td>
<td>Table 4.4 Multidimensional Matrix</td>
<td>28</td>
</tr>
<tr>
<td>8.</td>
<td>Table 4.5 Fact Table</td>
<td>30</td>
</tr>
<tr>
<td>9.</td>
<td>Table 5.1 Student</td>
<td>38</td>
</tr>
<tr>
<td>10.</td>
<td>Table 5.2 Student Course</td>
<td>38</td>
</tr>
<tr>
<td>11.</td>
<td>Table 5.3 Grading Elements</td>
<td>38</td>
</tr>
<tr>
<td>12.</td>
<td>Table 5.4 Department</td>
<td>38</td>
</tr>
<tr>
<td>13.</td>
<td>Table 5.5 Grade Functions</td>
<td>39</td>
</tr>
<tr>
<td>14.</td>
<td>Table 5.6 Faculty</td>
<td>39</td>
</tr>
<tr>
<td>15.</td>
<td>Table 5.7 ST Grade Elements</td>
<td>39</td>
</tr>
<tr>
<td>16.</td>
<td>Table 7.1 Test Cases</td>
<td>51</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Figure 2. 1 Use Case for User Logon</td>
<td>12</td>
</tr>
<tr>
<td>2.</td>
<td>Figure 2. 2 Use Case for User Interface</td>
<td>13</td>
</tr>
<tr>
<td>4.</td>
<td>Figure 2. 3 Use Case for Administrator</td>
<td>14</td>
</tr>
<tr>
<td>5.</td>
<td>Figure 4. 1 Dimensional Elements</td>
<td>25</td>
</tr>
<tr>
<td>6.</td>
<td>Figure 4. 2 Dimension Attributes</td>
<td>26</td>
</tr>
<tr>
<td>7.</td>
<td>Figure 4. 3 Star Schema</td>
<td>31</td>
</tr>
<tr>
<td>8.</td>
<td>Figure 4. 4 Snowflake Schema</td>
<td>32</td>
</tr>
<tr>
<td>9.</td>
<td>Figure 5. 1 System Architecture</td>
<td>34</td>
</tr>
<tr>
<td>10.</td>
<td>Figure 5. 2 Conceptual Model</td>
<td>36</td>
</tr>
<tr>
<td>11.</td>
<td>Figure 5. 3 Logical Model</td>
<td>37</td>
</tr>
<tr>
<td>12.</td>
<td>Figure 5. 4 Star Schema</td>
<td>40</td>
</tr>
<tr>
<td>13.</td>
<td>Figure 5. 5 Subject Interface</td>
<td>41</td>
</tr>
<tr>
<td>14.</td>
<td>Figure 5. 6 Subject Grade Elements</td>
<td>42</td>
</tr>
<tr>
<td>15.</td>
<td>Figure 5. 7 Subject Grade Rules</td>
<td>43</td>
</tr>
<tr>
<td>16.</td>
<td>Figure 5. 8 Class Interface</td>
<td>44</td>
</tr>
<tr>
<td>17.</td>
<td>Figure 5. 9 Faculty Interface</td>
<td>45</td>
</tr>
<tr>
<td>18.</td>
<td>Figure 5. 10 Student Interface</td>
<td>46</td>
</tr>
<tr>
<td>19.</td>
<td>Figure 5. 11 Reporting</td>
<td>47</td>
</tr>
<tr>
<td>20.</td>
<td>Figure 7. 1 New Subject Input</td>
<td>53</td>
</tr>
</tbody>
</table>
21. Figure 7.2 Edit Subject Input.........................................................54
22. Figure 7.3 Rules Input.................................................................55
23. Figure 7.4 Elements Input...........................................................56
24. Figure 7.5 Report...............................................................57
Chapter 1

INTRODUCTION

1.1 Objective

The objective of Rules based grading and reporting using Datamart is to design and develop an integrated student grade collection, calculation and reporting environment intended to remove the manual calculation processes involved in student grade determination.

Some University departments such as Department of Criminal Justice has very flexible rules in calculating the subject grades across different departments which results in complex manual calculations. The Student grade in a subject is based on different elements such as quiz scores, term paper, mid term, final exams each element has different weight in calculating the final student grade and process of calculating the final grade varies for each subject. Current tools such as Webct has limitations for example it is not flexible enough to allow the faculty to define different elements and the rules to calculate the grade from each grade element. The Webct also lack the detailed reporting or the historic comparison which can help the faculty to understand the effectiveness of different elements in calculation of student grades.
The project will demonstrate the value of data marts in the University environment, by providing easy access to historical information and trend analysis. It will also demonstrate the best practices in implementing the end user interface and transactional database for data capturing. The different data sources for data capturing for example web, excel files will be integrated into one system. The transactional database is designed for the quick input and manipulation of data

1.2 Purpose of this Project

The main purpose of this project is to develop a data mart and transactional system capable of storing, capturing and manipulating the student grades. The system then derives the metamorphoses of this data towards valuable and accurate knowledge retrieval for quicker, better analysis of student grades. The system provides several types of data access capabilities to retrieve and analyze the data contained in a data mart, on-line transaction processing. The following high level goals were established at the start of the project.

- Review the current grading calculation process.
- Design the interface to capture the student Grades.
- Provide the data mart for the analysis and reporting.
- Provide the enough flexibility in the web data interface to adjust the grading rules
- Format the Grades data such that it is easier to understand and to access.
• Deliver a set of pre-written reports.
• Provide a reporting tool that allows users to write their own adhoc reports.
• Implement the best practices for end user web design.
• Provide a reporting tool that allows users to write their own adhoc reports.
• Provide the users with and easy method to modify, save, & run their own reports and the pre-written reports.
• Provide training for users in the use of the front-end and in understanding the data structure.

1.3 Overview

The project implementation combines the System Development Life cycle and rapid application prototype project management disciplines and moves through defining the requirements, design, implementation, testing, acceptance and maintenance. The prototype is built at different project stages to further refine the requirements and design. The modular application design is used, in contrast to one monolithic chunk of tightly-coupled code. Every unit module interface directly with other modules. The modular design helps the further system extensions and keeps the dependencies between different code chunks clear. The project consists of following three high level modules, each module is then further distributed.
• Grade Rules Calculation and storage.
• Building a data mart
• Providing an end-user query/reporting front-end to the data mart.

Grading Rules storage module allows the end user to define the grading calculation elements (quiz, final exams etc) and the weight each associate in the calculation. The interface allows user to define the calculation formulas for each element and overall weight of each towards the final grade. Data Mart module provides the historical snapshots of student enrollment information as well as to perform trend analysis of the subjects. Reporting module provided the interface to canned reports as well as the adhoc reporting.

1.4 Project Methodology

The systems development life cycle (SDLC) conceptual model is used in project implementation. The project follows the different stages from an initial feasibility study through maintenance of the completed application. The waterfall model is used in the project implementation. The functional specifications are defined using the requirements gathering process. The project design is built on the basis of functional specification. The database modeling is done using the conceptual, logical, physical models, the star schema defines the dimensions used in the design. The wireframes are used to model the front end interface. The prototype are built at different phases to further refine the design and requirements thus the projects implementation methodology combines the System
development life cycle and rapid prototyping project management paradigms. The Table 1.1 lists the various project phases.

<table>
<thead>
<tr>
<th>Requirements Gathering</th>
<th>User Requirements assessment will collect the user and system requirements for project. User interview sessions will be conducted to understand the system need and establish overall goals of the project.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>In the planning phase project plan with schedule is developed.</td>
</tr>
<tr>
<td>Design</td>
<td>On the basis of requirements gathered and functional specification detailed design will be made. The Design will include the database, front end and other integration components.</td>
</tr>
<tr>
<td>Integration</td>
<td>All the designed modules will be integrated in one unit in this phase.</td>
</tr>
<tr>
<td>Testing</td>
<td>Test cases will be built in this phase to ensure the project quality.</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Project maintenance schedule is determined in this phase.</td>
</tr>
</tbody>
</table>

Table 1.1 Project Development Lifecycle
Chapter 2

USER REQUIREMENTS ASSESSMENT

2.1 Overview

The Software Requirements Analysis Process covers the complex task of eliciting and documenting the requirements of all the system users, modeling and analyzing these requirements and documenting them as a basis for system design. An effective requirements gathering process is the most critical driver of software project success. Getting the requirements right – and getting the right requirements – can mean the difference between a successful project – one that satisfies the needs of its users and is delivered on-time. The primary goal of the requirements gathering phase is to identify what constitutes as a success for this particular phase of the data warehouse project. In particular, end user reporting / analysis requirements are identified, and the project team will spend the remaining period of time trying to satisfy these requirements.

Based on the information gathered in the requirement gathering phase and the identification of user requirements a more concrete definition of other details such as hardware sizing information, training requirements, data source identification, and most importantly, a concrete project plan indicating the key milestones of the data mart project is prepared.
2.2 User Requirements Process

- The requirements gathering process started with requesting information on existing processes. Existing processes were studied, an interview sessions were done with the end users to understand the perspective on what the users of the system think they need. Through elaboration of the Use Cases, user requirements are thoroughly explored, fully understood and documented. The process started with a one on one session with the end user to understand the high level leader high-level business goals and project vision.

- The next principal use cases are identified, the use cases in a Use Case workshop. The Use case workshop results in a strong understanding of how the users interact with the system (people, roles, other systems, time) and the services they require from the system.

- Prototyping is done to explore the user interface features. The preliminary prototype represents user interface services to be provided by the system. The navigational components requirements were determined, along with a small amount of functionality to give the users a taste of the look-and-feel of their system implemented in a browser environment.

- The further analysis is done on the requirements to map the process, ensuring that we serve all the system goals.
• Non-user-interface Use Cases are explored. Non-user-interface Use Cases include file conversions, integration with the data or processes of other key systems, data uploads or downloads, batch processes, etc.

• Every software project has risks. Knowing what those risks are and planning to proactively

• The business users sign-off on business requirements.

• User participation in prototyping and Use Case exploration produces an interested, active extended team. By the end of the business

2.3 Requirements Areas

The table 2.1 lists the key requirements areas for this project. Each area requirements will be explored via user interview sessions and requirements are further refined by building prototypes at different stages of the project.

<table>
<thead>
<tr>
<th>Key Requirement Areas</th>
<th>The interface requirements for the system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>The interface requirements for the system</td>
</tr>
<tr>
<td>Security</td>
<td>Data encryption</td>
</tr>
<tr>
<td>Hardware</td>
<td>Hardware special requirements</td>
</tr>
<tr>
<td>User level</td>
<td>Roles for the users admin/faculty</td>
</tr>
<tr>
<td>Software System Maintenance</td>
<td>Database vendor to be used.</td>
</tr>
<tr>
<td>System Maintenance</td>
<td>Maintenance needs</td>
</tr>
<tr>
<td>Error Handling</td>
<td>System response on errors</td>
</tr>
</tbody>
</table>

Table 2.1 Key Requirement Areas
2.4 Requirements Questions/Answers

The requirement gathering process for the project started with interview session with Dr Xin Ren. Following questions are asked to understand the overall goals of the system and identify the initial requirements.

1. What are your goals in developing this system?
   The goal is to reduce the manual calculations needed to calculate the student grades and also provide an effective reporting. The automated calculation system should provide the easy way to change the grading rules for example user should be able to configure whether top three or top four quiz scores should be used to calculate the overall grade for the student.

2. Who are the key users of the system?
   The key users in the starting are the faculty staff but system should support future extensions where a student can log on to the system and analyze the grade and all the elements for example student can run the analysis on how much he need to score for the next quiz in order to get a better Grade.

3. What is the most important business goal for this project?
   Flexibility to change the Grading rules and reporting.

4. Will the system change the way you are doing things now?
It will save lot of time and reduce the workload, the manual grade calculation process results in errors and it takes time and resources to validate. The system can increase further efficiency by providing a interface for the score uploads which often are recorded in different excel files. This will make sure that all the tests, quiz, report scores are in one place. This will also give the faculty insight into call performance at particular point in time and help the students.

5. Will the system help you be more efficient?
   Yes, it will help us to be more efficient. The further system extensions should also allow student logon where students can check the current grade directly, this will further help the faculty members to provide feedback to students.

6. What are the system deliverables?
   Grade calculation system and interface where grades can be uploaded from excel file. The system should allow the uploaded grades to associate with class and student. It should store the all tests, quiz scores for a class. The flexibility to change the calculation function is very important for example selection of top three quiz scores out of six quiz tests. The system should report how the change in calculation function changes the overall grades for the class.

6. Which reports do you need from the system?
System should report number of students with A grade etc and plot the bell curve. System should also help in historical analysis, users should be able to compare the class performances over the period of time. It should also help in analysis of grading elements for example how the change in rules have caused the change in class performances for the specified time periods.

7. What user roles should system support?

The system primary audience is faculty members but it should support student role extension. System will store student identification and this data needs to secure and should not be visible to everyone.

8. Any specific requirements on the user Interface?

System should be web based, the performance is key here. The data upload should not take too long and also it should support all common browsers. There should be a validation built within the system to prevent common user errors. The user interface should be simple and well organized. The reporting interface should support filters like time etc.

9. Any other requirements?

System should have a inbuilt help system which should guide the users in case something is not clear on the user interface. The help system should cover common errors and feedback. It should also include the browser issues for
example in they need to install a software plug in to get the site working. And that

2.5 Use Cases

Use Case modeling captures system's behaviors and requirements from end-user's point of view. Use Case detail documents scenario-driven story with flow-of-events. The use cases are discussed with the end users to further refine the requirements.

![Use Case Diagram](image)

**Figure 2.1 Use Case for User Logon**

Figure 2.1 shows the initial interaction in identifying the roles after the initial user logon. The user role could be faculty, administrator or student. The administrator is a special
role kept in the system, administrator can change the system settings for example
changing the location of file uploads or database settings. The Faculty role will be used
by all staff members for updating the classes, grades etc.

Figure 2.2 Use Case for User Interface
The figure 2.3 shows the common system functions for the faculty, the faculty will have option to import grades and import students. The import will support the excel upload to make the process more efficient. The Faculty can also define the grading rules and subject information.

![Diagram](image)

Figure 2.3 Use Case for Administrator

Figure 2.3 shows the administrator function in the system, administrator can change setups and add users to the security. Administrator can also add the look up values for various other function for example defining the subject elements look up which consists of values like quiz, reports, mid term etc.
2.6 Functional Specifications

The functional specifications for the system are collected via user interview sessions and use cases analysis.

Security:

- System should support multiple user security roles, the administrator role will allow the change in system set ups and add more users to the system. Faculty user roles will allow the staff members to add classes and subjects.
- Users should log on using username/password. System will have forgot password functionality.
- The system will have idle session time out to prevent any unauthorized use.
- The Student Id will be encrypted in the database.

Interface

- The data for students will be uploaded using the excel file, system will generate the upload template.
- Data for grades will be uploaded using the excel file, system will generate the upload template for the grades data bases on rules defined.
- The system should have interface for defining the Grade rules.
- Users should be able to define the rules using the pre programmed functions following functions will be supported top, max, min, average.
• System will allow the grade calculation at any point in time.
• System will consolidate all the data uploads to the related subject.
• System will support the drill down on the records

Reporting

• The reporting interface should show allow to save the custom reports
• The interface should have pre canned common reports.
• The most important report is to find out the number of students with specific grades.
• The time filter should allow the historical analysis of the class.
• The drill down should be available on all the reports for example the analysis of student should list all the grade elements.
• The common graph types should be supported for example bar graph to compare the student by grade.

Error Handling

• System should report the errors in a user friendly way for example selecting the wrong time filters.
• The system should check the browser compatibility and report to user if any plug-in needs to be installed.
• The all errors should be stored in the even logs for analysis by administrator.
Portals:

- The system should support the portal concept where all common/saved reports can be placed on the home page.

The functional specifications generated in the requirements gathering phase are further revised by building prototypes for every component.
Chapter 3

PROJECT PLAN

3.1 Project Tasks Identification and Deliverables

The section lists the tasks involved in the Rule Based Grading and Datamart implementation.

- Requirement Assessment: The initial requirements were gathered by user interviews at the end of this task functional specification for the project were made. The requirements were further refined during the prototypes review.
  Task Deliverable: Functional specification for Student Datamart.

- Physical Environment Setup: Based on the requirements identification initial physical environments were determined. Mysql database environment features were studied and it’s usefulness in the datamart environment.
  Task Deliverable: Database vendor to use and physical environment.

- Data Modeling: Database modeling is done on based of requirements to generate the conceptual, logical, physical model. The dimensional data model was made and star schema was defined.
  Task Deliverable: Conceptual model, logical model, physical model, star schema

- ETL: Extraction, Transformation logic is defined in this task. The transformation mappings from transactional database to dimensional database are defined. The facts level summary calculations are done.
- Frond end design was done using the use cases on basis of requirements. Different screen designs were done to capture the transactional data.

Task Deliverable: Front End user input screens.

- Report Development: Reporting interface is designed/developed in this task on basis of user requirements.

Task Deliverable: Reporting interface

- Rolling out to Production: The Production roll out and maintenance/performance tuning is planned in this step.

3.2 Project Tasks Schedule Plan

<table>
<thead>
<tr>
<th>Task List</th>
<th>Time estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Requirements Gathering</td>
<td>2 weeks</td>
</tr>
<tr>
<td>Use cases for initial requirements</td>
<td>1 week</td>
</tr>
<tr>
<td>Functional specifications indentified</td>
<td>1 week</td>
</tr>
<tr>
<td>Conceptual data model for Transactional database</td>
<td>2 week</td>
</tr>
<tr>
<td>Physical model for transactional database</td>
<td>2 week</td>
</tr>
<tr>
<td>Dimensional model and star schema</td>
<td>2 week</td>
</tr>
<tr>
<td>Front end wireframes</td>
<td>4 week</td>
</tr>
<tr>
<td>Prototype</td>
<td>3 weeks</td>
</tr>
<tr>
<td>Revised requirements on basis of prototype review</td>
<td>2 week</td>
</tr>
<tr>
<td>ETL design</td>
<td>3 weeks</td>
</tr>
<tr>
<td>Reporting design</td>
<td>3 weeks</td>
</tr>
<tr>
<td>Working System review</td>
<td>3 weeks</td>
</tr>
<tr>
<td>Revised the design on basis of system review</td>
<td>2 weeks</td>
</tr>
<tr>
<td>Test cases identification</td>
<td>1 week</td>
</tr>
<tr>
<td>Testing and bug fixes</td>
<td>1 week</td>
</tr>
<tr>
<td>Production deployment and maintenance</td>
<td>2 week</td>
</tr>
</tbody>
</table>

Table 3.1 Project Schedule
The project plan identifies the key tasks and time estimates needed to implement this project. The initial time estimates accuracy is analyzed in the conclusion section of this report to give the future researchers an idea on the effort magnitude involved in similar project implementations.
Chapter 4

DATAMART FOR GRADING SYSTEM

4.1 Datamart

This chapter lists the various datamart concepts to help in researching the right data model for this project. The various examples studied are analyzed in context to this project. Relational databases typically are optimized for online transaction processing systems. They are designed to meet the day-to-day operational system needs, and the database performance is tuned for those operational needs. Consequently, the database can retrieve a small number of records quickly, but it can be slow if you need to retrieve a large number of records and summarize data on the fly. For example, if we are building a grade trend report for a particular course for last five years, the data in the transactional systems may not be consistent because it is always changing. In addition, the database structure may be complicated for reporting [1].

Datamart is a database containing data that is integrated, aggregated and structure so that it can be used to support the reporting analysis and decision making process. Datamart are geared to help in the decision making process for example if we need to find out how the change in course material has changed the student performance in comparison to last years. Datamart also consolidates the data as there may be multiple transactional systems to collect the data, Datamart consolidates all the data into a single entity which helps in overall making decision process as all the data points are at one place.
Datamarts are designed for quick retrieval of the data where access path is not known in advance. Information is derived from other data by rolling up data into summaries, drilling down to get more detail or looking for patterns and trends.

4.2 Relational Database Design vs Datamart Design

The relational data model is designed into tables with columns and rows. Unique value such as student id is stored in multiple tables to identify the row. The relational model was much easier to understand, the language used to access the database did not require knowledge of how the data is physically stored. The initial release of relational databases also provided the adhoc query, reporting and analysis tool. A limited amount of history is retained in the relational model for example it is easy to determine grade of a particular student at current point in time but the analysis of final grades for all student in past five years may not be possible to do in easy manner. The relational models were normalized to make sure that redundant data is stored at one place. Normalization ensures that tables are correctly formed by putting related data together in one table and eliminating the redundancy. The one copy of data ensures that update anomalies are avoided and consistency is maintained [2].

The datamart are designed to optimize the performance of the querying of the data, the primary goal here is to query the data in a effective way without affecting the performance of transaction system and helps the data analysis for decision making process. The dimensional data model introduced the concepts of normalization,
redundancy. The model is built in way so users can understand the data, the star schema
design facilitate the Online Analytical processing.

Dimensional data model is different from the normalization forms as same data is
stored in a different way than third normal form which is used for Online transaction
(OLTP) systems. The Online transaction systems are process oriented for example
recording the student enrollments but the dimensional database design is more subject
oriented which is designed to answer specific questions for example how many students
have received A grade in a particular course. The dimensional database is designed to
support the analysis and projections, this is also know as Online Analytical processing
database(OLAP). The data is typically loaded in batches in the Datamart as it may come
from different sources. The use of dimension solves number of issues related to query
run time, a query to calculate five years of grade average for students in a subject may
take hours to run in transational database but same can be run in shorter time in the
dimensional model. The table below lists the key differences in the OLTP and OLAP
databases [4].

<table>
<thead>
<tr>
<th>OLTP Databases</th>
<th>OLAP Databases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data is atomized</td>
<td>Data is summarized</td>
</tr>
<tr>
<td>Data is current</td>
<td>Data is historical</td>
</tr>
<tr>
<td>Processes one record at a time</td>
<td>Processes many records at a time</td>
</tr>
<tr>
<td>Process oriented</td>
<td>Subject oriented</td>
</tr>
</tbody>
</table>

Table 4.1 OLTP vs OLAP Database
4.3 Dimensional Data Modeling Key Concepts

Dimensional data model views and models the data from a different perspective, instead of considering and entity which represents a thing such a student or a relationship between those entities, a dimensional data model describes data using dimensions and facts [7]. Dimension is a category of information for example time dimension. Dimension attributes are level within the dimension which further categorizes it for example, month is a attribute of time dimension. Hierarchy is the relationship between different attributes in a dimension for example, the hierarchy for time dimension is Year → Quarter → Month → Day. The fact table contains the factual information in the datamart and is usually the largest table. The fact table is where all detail data is stored for example the student grades for a particular subject in last five years.

Each dimension indentified in the data model is implemented in the database as dimension table. Dimensions are the qualifiers that make the measures of the fact table meaningful. Whether data is dimension or fact can be found out by answering questions like Is the data static and data describes something. Dimensions like Student Id do not change frequently whereas fact table will store the summary data on student grades. The fact table will be the largest table in the datamart, the dimensional table have limited data. The dimensional table typical contains more text fields whereas fact table contains more numeric fields. Common example of factual data listed below

- how many students scored A grade in a subject
• What was the comparison of Grades in a subject in two academic terms
• Trending of the grades year over year

Each set of questions represents the factual data.

4.4 Dimensional Modeling

A dimension can define multiple dimension elements for different levels of summation. For example, all the elements that relate to the structure of a time dimension comprise of more dimensions. Dimensions are made up of hierarchies of common elements because of hierarchy involved in the dimensions, the queries can be constructed to access data at higher level or lower level. The process of aggregating data at higher level is called rollup and drill down for summarizing at lower level. Figure 4.1 below shows the time dimension

![Dimensional Elements Diagram](image)
4.5 Dimension Attributes

A dimensional attribute is a column in a dimension table. The attributes describes the level of summary within the hierarchy. The attributes describe the dimensional elements in terms more understanding to users whereas the dimension elements define the hierarchical relationships within a dimension table. Dimension attributes describe the items in dimensions so they are most useful in text. Sometimes during the design process it is unclear whether a numeric data field from a data source is a measured fact or an attribute. Generally, if the numeric data field is a measurement that changes each time we sample it, it is a fact.

![Figure 4.2 Dimension Attributes](image)

4.6 Dimension Tables

Designing using the dimension model involves many dimension tables but there may only be one or few fact tables. The dimension table is a reference to fact table where
descriptions and more static information about a piece of data are kept. A dimension table stores the textual descriptions of the dimensions of the business. A dimension table contains an element and an attribute, if appropriate, for each level in the hierarchy. The lowest level of detail that is required for data analysis determines the lowest level in the hierarchy. Levels higher than this base level store redundant data. This denormalized table reduces the number of joins that are required for a query and makes it easier for users to query at higher levels and then drill down to lower levels of detail. Traditional relational databases are organized around a list of records. Each record contains related information that is organized into attributes (fields). The typical student table in a database contains fields for name, email, address etc. While this table has several fields of information, each row in the table pertains to only one student. Each record has a one to one correspondence [8]. The example table is show below.

<table>
<thead>
<tr>
<th>Student Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Vikram</td>
</tr>
<tr>
<td>Smith</td>
</tr>
<tr>
<td>Bill</td>
</tr>
</tbody>
</table>

Table 4.2 Relational Data
Any combination of fields from the student table in the matrix will always give a one-to-one correspondence, this shows that this table is not multidimensional and would not be well suited for a dimensional database. However, consider a relational table that contains more than a one-to-one correspondence between the fields of the table. Suppose you create a table that contains grade data for students in different years.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Grade</th>
<th>Number of students</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSC 177</td>
<td>A</td>
<td>23</td>
</tr>
<tr>
<td>CSC 189</td>
<td>A</td>
<td>13</td>
</tr>
<tr>
<td>CSC 177</td>
<td>B</td>
<td>45</td>
</tr>
</tbody>
</table>

Table 4.3 Multidimensional Data

The table 4.3 itself to multidimensional representation because it has more than one grade per subject. The table 4.4 shows a two-dimensional matrix that better represents the many-to-many relationship of product and region data.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Grade</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSC177</td>
<td>23</td>
<td>22</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>CSC189</td>
<td>23</td>
<td>55</td>
<td>17</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4 Multidimensional Matrix
There are numerous advantages of the dimensional table over the traditional relational table. A dimensional approach simplifies access to the data that you want to summarize or compare. For example, if you use the dimensional table to query the number of A Grades in all Subjects, the database just aggregates the total of all row values in that column. To perform the same query on the relational table, the database server has to search and retrieve each row where the Grade column equals A and then aggregates data. In queries of this kind, the dimensional table can total all values of the Grade column in a fraction of the time it takes the relational table to find all the A Grade records.

4.7 Fact Table

The fact table contains factual information, and it usually the largest table in the data warehouse and is fastest growing. The fact tables are typically where all the detailed data is stored for example all student grades. The information in the fact table does not have to be detailed, it could also be the summarized data, such as total number of A graded in a particular subject. The fact table stores the measures of the dimensions and points to the key value at the lowest level of each dimension table. The measures can be quantitative or factual data about the subject. The measures are generally numeric and correspond to the how much or how many aspects of a question. A measure can be based on a column in a table or it can be calculated. The table shows a fact table whose measures number of Grades in a particular year.
Before you design a fact table, you must determine the granularity of the fact table. The level at which data is kept in the fact table is known as granularity of the table. How you define an individual low-level record in that fact table. The granularity might be the individual transaction, a daily snapshot, or a monthly snapshot. The fact table above one row for every year. Thus, the granularity of the fact table is expressed as grades per year.

The lookup table provides the detailed information about the attributes. For example, the lookup table for the Quarter attribute would include a list of all of the quarters available in the data mart. Each row (each quarter) may have several fields, one for the unique ID that identifies the quarter, and one or more additional fields that specifies how that particular quarter is represented on a report (for example, first quarter of 2001 may be represented as "Q1 2001" or "2001 Q1").
A dimensional model includes fact tables and lookup tables. Fact tables connect to one or more lookup tables, but fact tables do not have direct relationships to one another. Dimensions and hierarchies are represented by lookup tables. Attributes are the non-key columns in the lookup tables.

4.8 Star Schema

In the star schema design, the fact table sits in the middle and is connected to other dimensions. Each dimension is represented as a single table. The primary key in each dimension table is related to a foreign key in the fact table. All measures in the fact table are related to all the dimensions that fact table is related to. In other words, they all have the same level of granularity. A star schema can be simple or complex. A simple star consists of one fact table; a complex star can have more than one fact table [6].

Figure 4. 3 Star Schema
4.9 Snowflake Schema

In snowflake schema, that dimensional table is normalized into multiple lookup tables, each representing a level in the dimensional hierarchy.

![Snowflake Schema Diagram](image)

**Figure 4.4 Snowflake Schema**

The main advantage of the snowflake schema is the improvement in query performance due to minimized disk storage requirements and joining smaller lookup tables. The main disadvantage of the snowflake schema is the additional maintenance efforts needed due to the increase number of lookup tables.
Chapter 5

SYSTEM DESIGN

The chapter illustrates the system design. The transactional database design is modeled using the conceptual and logical models. The data mart is modeled using dimensional modeling. The goal is to organize the data in a standard relational structure that optimizes reporting and analysis capabilities. Each dimensional model is made up of a fact table and a set dimension tables. The fact table has a compound key that is made up of the primary keys from each of its associated dimension tables.

5.1 System Architecture

The modular application design is used, in contrast to one monolithic chunk of tightly-coupled code. Every unit module interface directly with other modules. The modular design helps the further system extensions and keeps the dependencies between different modules clear.

5.1.1 Grade Rules Storage Module

Grading Rules storage module will allow the end user to define the grading calculation elements (quiz, final exams etc) and the weight each associate in the calculation. The interface will allow user to define the calculation formulas for each
element and overall  The Data Capture interface will allow the user an easy capture of the data for different data elements. It will give the option of bulk upload from an Excel file. For example, an Excel file for the quiz scores will be input directly into the system.

![Figure 5.1 System Architecture](image)

5.1.2 Data Capture Module

Grading Rules data capture will allow the end user to define the grading calculation elements (quiz, final exams, etc.) and the weight each associate in the calculation. The interface will allow user to define the calculation formulas for each element and overall. The Data Capture interface will allow the user an easy capture of the data for different
data elements. It will give the option of bulk upload from a excel file for example excel file for the quiz scores will be input directly into system.

5.1.3 Grade Calculation Module

Grade Calculation module will allow the calculation of Student grade on the basis of Rules and captured data. It will have the error checks to guide the user in case of missing data for example flags the missing scores of a student. The calculation module will have a dynamic interaction with the Rules Storage module which will give the user to change the rules anytime if need to and calculation will be adjusted accordingly.

5.1.4 Data Reporting Module

Data mart reporting will allow the user to view the historical class performances and compare the effectiveness of different rules. The reporting module also provides the way to save reports. The user can save the reporting filters.

5.2 Database Modeling

The data modeling identifies the entity types. Data attributes are assigned to entity types and the associations between them.
5.2.1 Conceptual Model

The figure 5.2 shows the conceptual model for the database, it shows the primary keys used and foreign key relationships.

5.2.2 Logical Model

The figure 5.3 shows the logical model for the database, it shows the primary keys used and foreign key relationships.
Figure 5.3 Logical Model
5.2.3 Physical Data Model

<table>
<thead>
<tr>
<th>Table 5.1 Student</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Student</strong></td>
</tr>
<tr>
<td>Student_id</td>
</tr>
<tr>
<td>First_name</td>
</tr>
<tr>
<td>Last_name</td>
</tr>
<tr>
<td>Status</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5.2 Student Course</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Student_course</strong></td>
</tr>
<tr>
<td>Student_course_id</td>
</tr>
<tr>
<td>Course_id</td>
</tr>
<tr>
<td>Student_id</td>
</tr>
<tr>
<td>Grade</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5.3 Grading Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Grading_elements</strong></td>
</tr>
<tr>
<td>Element_id</td>
</tr>
<tr>
<td>Course_id</td>
</tr>
<tr>
<td>Function_id</td>
</tr>
<tr>
<td>Element_date</td>
</tr>
<tr>
<td>Status</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5.4 Department</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Department</strong></td>
</tr>
<tr>
<td>Department_id</td>
</tr>
<tr>
<td>Name</td>
</tr>
</tbody>
</table>
The physical model lists the physical attributes for the logical model.
5.2.4 Datamart Star Schema

Figure 5.4 Star Schema
5.3 Front End Wireframes

The wireframes captures the design elements of user interface. The end user screens are divided into left, middle navigation panes. The left pane contains the all higher level objects. The center pane is further divided into top and bottom panes. The top pane contains the parent object associated with left navigation while the child object is in bottom pane.

5.3.1 Subject Interface

<table>
<thead>
<tr>
<th>Subjects</th>
<th>New</th>
<th>Edit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Faculty</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Student</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reporting</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Grade Elements**

<table>
<thead>
<tr>
<th>Semester</th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Admin</th>
<th>Elements</th>
<th>Users</th>
<th>Help</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Element</th>
<th>Function</th>
<th>Six</th>
<th>Max score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final</td>
<td>none</td>
<td>1</td>
<td>200</td>
</tr>
<tr>
<td>Report</td>
<td>none</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Quiz</td>
<td>top</td>
<td>3</td>
<td>20</td>
</tr>
</tbody>
</table>

Figure 5.5 Subject Interface
The subject interface is divided into new and edit functionality, end user can create a new course. The department selection and faculty selections are dropdowns populated from department, faculty objects. The subject screen also have brief description. The bottom center pane contains the Grade elements and Rules child objects for a subject. The Grade elements are associated with a subject every semester this makes sure that historical rules are maintained for further analysis. The Grade element child object has element name which are populated from the element object, the function selections are
predefined for example top function selects the top scores out of set of scores. The max score records the maximum possible score in a subject. The figure 5.3 shows the rules interface.

![Figure 5.7 Subject Grade Rules](image)

The rules are defined for the Grades in the rules interface. The upper and lower limit gives the flexibility end users to define grade ranges. The rules are tied to semester as this makes sure the historical changes are recorded.
5.3.2 Class Interface

This class interface in Figure 5.8 shows the class object interface design, the subject, semester and faculty dropdowns are populated from subject and faculty objects. The bottom pane contains the student object details for a class. The students details can be imported via excel file. The student grades can also be imported via excel file. The system will automatically generate the excel template files for grades collection depending upon the rules defined for a course for example if the grade elements for a
subject has three quiz scores then the excel file contain the three columns for grade collection. The idea is to provide the template files to the end users and let them upload the file at any point of time, this gives the flexibility of offline data collection.

5.3.3 Faculty Interface

![Faculty Interface Diagram](image-url)

Figure 5.9 Faculty Interface
The faculty interface captures the details of faculty, the bottom pane list the all subjects which faulty is teaching. The drill down on the class grades will allow the easy way to see the class performance.

5.3.4 Student Interface

![Student Interface Diagram]

Figure 5.10 Student Interface

The student interface captures the details of student, this is again a quick way to browse the data for a student.
The search interface will allow the search on different objects and also will let you to save the searches. The saved searches are tied back to the user portals.

5.3.5 Reporting Interface

![Diagram of Reporting Interface]

Figure 5.11 Reporting
The reporting interface top pane allows the user to select filters, the bottom pane is divided into the grid view and graph view. The grid allows the further dropdown on the details.
6.1 Loading the Data into Tables

The extract process involves extracting the data from the source systems. The data is consolidated from different source systems. The transform stage transforms the data extracted by applying series of rules or functions. The transformation functions can be complex or very simple. The load phase loads the data into the target database. The data is loaded into the different tables for student datamart.

6.2 Student Dimension

The student dimension is populated by selecting all students who are enrolled into a course. The student registration details are also taken from the registration table and student major table. The other details of student are also populated like first name, last name etc. The simple query to load the student dimension is show below.

```
INSERT INTO [TABLE_NAME] (SELECT [ATTRIBUTE, [ATTRIBUTE_LIST]] from [TABLE_NAME] WHERE [CONDITION])
```

The student dimension will also have a active flag which will be part of the transformation logic to identify if the student has no courses enrolled and thus is inactive.

6.3 Course Dimension

The course dimension lists all the courses, it is populated using the distinct course_id from the course transaction table, the other attributes of course is also stored.
6.4 Grades Rules Dimension

Grades dimension contains the Grading rules for each course, the dimension is populated by taking the distinct on rules_id from the grading rules table. This will indicate the rules involved in calculating the grade of course.

6.5 Grading Elements Dimension

The grading elements dimension stores the grading elements involved in a course, it will be populated from the Grading elements table and will contain the element_id. The element id will help in the drill down of data to specific grade elements. Using the element_id each score in the element can be determined.

6.6 Time Dimension

Time dimension contains all the possible dates for a certain amount of time and some other summarized values like day of the week or quarter of year, etc. for each date. The procedure to upload the time dimensions is shown in appendix A.
Effective testing is key to the project successful implementation. The project follows the unit testing and black box testing methodologies. The test data set is made working with the end users. The test excel upload files were used to check the upload process and if the data is being populated correctly. The following table defines the test cases and how the results were documented.

<table>
<thead>
<tr>
<th>Test#</th>
<th>Test Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log in screen, check if the login fails in case wrong username/password is given.</td>
</tr>
<tr>
<td>2.</td>
<td>Forgot password sends the email out to the user.</td>
</tr>
<tr>
<td>3.</td>
<td>Login using the right username/password</td>
</tr>
<tr>
<td>4</td>
<td>Does the screen render properly on initial logon? The screen should not cut off on resolutions greater than 1024 x 768</td>
</tr>
</tbody>
</table>

Table 7.1 Test Cases
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5.</td>
<td>Check the permissions with different logon. The administrator should see more options for example the admin controls.</td>
</tr>
<tr>
<td>6.</td>
<td>Create a new subject using the subjects interface.</td>
</tr>
<tr>
<td>7</td>
<td>Create the new grading elements and associate those to the subject.</td>
</tr>
<tr>
<td>8</td>
<td>Create the new grading Rules.</td>
</tr>
<tr>
<td>9</td>
<td>Create new class by selecting the subject and faculty.</td>
</tr>
<tr>
<td>10</td>
<td>Import the class students from the excel file.</td>
</tr>
<tr>
<td>11</td>
<td>Review the excel template for grades upload generated by the system.</td>
</tr>
<tr>
<td>12</td>
<td>Create new Faculty</td>
</tr>
<tr>
<td>13</td>
<td>Run the reports by selecting time filters.</td>
</tr>
</tbody>
</table>

Table 7.1 (continued)

7.2 Sample Run

The screen captured in this section shows the sample system run. The left hand navigation represents the objects for data capture for example Subject, Grade, Class. The left navigation also has the links to run reports and perform system search. The Sample run starts with capturing the subject details. The subject stores all information related to the coursework. The next step is to capture the faculty details. The class object builds
the relation between subject, Faculty and student. The Sample run also demonstrates the report generated for a class.

Figure 7.1 New Subject Input

The new subject input screen is used to create the new subject, it takes the Subject Name, Subject Description, Subject Faculty as the input. The Subject Name is a required field. The Subject object stores the detail of the course work.
The edit subject input screen is used to edit the existing subjects, it takes the Subject Name, Subject Description, Subject Faculty as the input. The Subject Name is a required field. The screen also has the links to define the template Update and Rules for the Subject object.
Figure 7.3 Rules Input

The Rules screen allows the faculty to enter the grading rules. The Rules are stored per term this allows the rules to be changed every semester and provided enough flexibility. The maximum and minimum marks for the subject and corresponding Grade for that range is also defined.
Figure 7.4 Elements Input

The interface allows user to enter the different elements for a subject. The elements are associated with a term this provides enough flexibility for the change of rules. The maximum score for the element is defined as well as the count. The function dropdown is used to select the function needed to select the scores for example top five quiz scores out of total 10.
Figure 7.5 Report

The report for a sample class is show, the grades for the class can be uploaded at any point during the semester term and overall class performance thus can be calculated. The Figure 7.6 shows the class performance in terms of A and B Grades.
Chapter 8

CONCLUSION

The implementation of data mart enabled users to gain faster access to common data utilizing a technique called dimensional data modeling, which optimizes data for reports. For example, since data is prepared in common format, users with little or not training at all, can browse a data mart and obtain information as needed. Data marts improved the end user response time, as it contains raw data which allows computer systems to focus on a single task, thus, improving performance.

Data marts are the "corner stores" of the enterprise these days, the primary goal of this project is to identify and demonstrate the steps to build an effective enterprise transactional and datamart system. Building the transactional system and datamart in the project helped to understand the complete lifecycle of enterprise data and also highlights how the new transactional systems should be designed for effective reporting. The Rules based grading system also gives a flexible tool to the Faculty members to understand the effectiveness of various grading elements and helps them to analyze the class performance at any point of time.

The future extension of this system can allow students to log into the system and give them the snapshot of there performance in a particular course at any point of time rather than to wait till end of the semester, it can also provide a guidance to them on how
they need to score in the upcoming tests to get a better grade. The future extensions can also research on optimizing the data loads for the data mart. The data load is build in a current project as a batch job which runs every night but real time data synch can be helpful to avoid any delays.
APPENDIX A

MySQL

MySQL is the leading open source database solution used today to power online enterprise, embedded, and business intelligence applications. For over twelve years, the MySQL database server has been the heart of database systems that serve a growing and intensely demanding customer base. The “M” in the LAMP stack (Linux, Apache, MySQL, PHP/Perl/Python), MySQL has been battle-tested by heavy transaction processing applications, terabyte-sized data warehouses, and high-traffic Web sites, and found to be the proven leader in open source database technology. Thousands of well-known companies such as Google, Yahoo, NY Times, Cox Communications, The Associated Press, Symantec, Alcatel, Nokia, Nortel, Cisco, Zappos, and others rely on MySQL to manage their data-driven applications.

The same MySQL server that exceeds expectations in these environments is the same database that’s also used to manage the information needs of small-medium applications that rely on a bundled database as well as deeply embedded systems that demand a high-performing and reliable database with a very small footprint. Having proven itself in the bleeding-edge world of technology start-up’s, Web 2.0 and other such forward-thinking companies.
MySQL contains a solid core feature set that is suitable for all data warehousing use cases. The following are just some of the features in the MySQL database server that help enable data warehousing:

• Data/Index partitioning – available in MySQL 5.1 and higher; supports range, hash, key, list, and composite partitioning. Partition “pruning” is available, which involves MySQL only examining the partitions it needs to satisfy a particular query instead of an entire table or index. Partition management is also supported (ADD PARTITION, DROP PARTITION, etc.)

• No practical storage limits – for example, 1 tablespace=110TB limit

• Automatic storage management – autogrowth data files, etc.

• ANSI-SQL support for all datatypes – including BLOB and XML

• Built-in Replication – simple and easy to configure

• Main memory tables – keeps all data in-resident in RAM; perfect for dimension tables

• Support for a variety of indexes – B-tree, fulltext, clustered, hash, GIS

• Multiple-configurable data/index caches

• Pre-loading of index data into index caches

• Parallel data load – loads multiple files at the same time

• Multi-insert DML – allows array-style processing via normal INSERT commands

• Data compression – provides enormous storage savings

• Read-only tables – protects sensitive data

• Encryption – further protection for sensitive data

• Cost-based optimizer – eliminates need for rule-based query writing
- Wide platform support – no need for special hardware or operating systems
APPENDIX B

Project Code

// The class defines the common package for error logging as well as utilities to connect to the
Database.

package com.csus.ctmgmt.common;

public class InterfacesException extends Exception
{
    
    String strPrefixMsg;
    String strErrorCode;
    String strSuffixMsg;
    Exception wrapException;

    /**
    * Constructor sets the Prefix, ErrorCode and Suffix
    * @param strPrefix prefix of error message
    * @param strErrorCode error code
    * @param strSuffix suffix of error message
    */

    public InterfacesException(Exception wrapException, String strErrorCode, String strSuffix)
    {
        super(wrapException.toString());
        this.wrapException=wrapException;
        this.strErrorCode=strErrorCode;
        this.strSuffixMsg=strSuffix;
    }

    public InterfacesException(String strPrefix, String strErrorCode, String strSuffix)
    {
        this.strPrefixMsg=strPrefix;
        this.strErrorCode=strErrorCode;
        this.strSuffixMsg=strSuffix;
    }

    /**
    * Constructor sets ErrorCode and Suffix
    * @param strErrorCode error code
    */

    public InterfacesException(String strPrefix, 
                        String strErrorCode, String strSuffix) {
        this.strPrefixMsg=strPrefix;
        this.strErrorCode=strErrorCode;
        this.strSuffixMsg=strSuffix;
    }
* @param strSuffix suffix of error message
*/
public InterfacesException(String strErrorCode,String strSuffix) {
    this.strErrorCode=strErrorCode;
    this.strSuffixMsg=strSuffix;
    this.strPrefixMsg="";
}
/**
 * Constructor sets ErrorCode
 * @param strErrorCode error code
 */
public InterfacesException(String strErrorCode) {
    this.strErrorCode=strErrorCode;
    this.strSuffixMsg="";
    this.strPrefixMsg="";
}
/**
 * No argument constructor
 */
public InterfacesException() {
    strPrefixMsg="";
    strErrorCode="";
    strSuffixMsg="";
}
/**
 * Getter Methods
 * @return strPrefixMsg prefix message
 */
public String getPrefix() {
    return strPrefixMsg;
}
/**
 * Getter Methods
 * @return strErrorCode error message
 */
public String getErrorCode() {
    return strErrorCode;
}
/**
 * Getter Methods
 * @return strSuffixMsg prefix message
 */
public String getSuffix() {
public void rethrow() throws Exception {
    throw wrapException;
}

package com.csus.ctmgmt.common;

/*
 * Modification Log
 * ----------------------------------------------------------------------
 * Ver  Date    Modified By     Description
 * ----------------------------------------------------------------------
 * /

import java.lang.*;
import java.io.*;
import java.text.*;

public class Logger {

    private static String ERROR_LOG_FILE = null;
    private static String DEBUG_LOG_FILE = null;
    private static String AUDIT_LOG_FILE = null;
    private static String PERFORMANCE_LOG_FILE = null;
    private static String DEBUG_LOG_LEVEL = null;
    private static String AUDIT_LOG_LEVEL = null;
    private static String PERFORMANCE_LOG_LEVEL = null;
    private static String strAppendAtStart = null;
    private static String strAppendFinally = null;
    private static String strErrorCode  = null;
    private static String TAB    = "\t";

    private static PrintWriter out =null;

public static void handleException(String componentName, InterfacesException iExc) throws Exception {
    String strLogMessage = null;
    String strErrorMsg = null;

    try {

        PropertyHandler properties = new PropertyHandler("Global");

        // Get the error code, suffix and prefix from the exception object
        strErrorCode = iExc.getErrorCode();
        strAppendAtStart = iExc.getPrefix();
        strAppendFinally = iExc.getSuffix();

        ERROR_LOG_FILE = properties.getStringValue("error_log");
        strErrorMsg = properties.getStringValue(strErrorCode);

        strLogMessage = strAppendAtStart + TAB+ strErrorCode + TAB+ strErrorMsg + TAB+ strAppendFinally;

        System.out.println(strErrorCode);
        System.out.println(strAppendAtStart);
        System.out.println(strAppendFinally);

        writeToFile(componentName, ERROR_LOG_FILE, strLogMessage);
        // System.exit(1);
    } catch(InterfacesException ieExc) {
        debugMsg(componentName, ieExc.getPrefix() + ieExc.getErrorCode() + ieExc.getSuffix());
    }
}

public static void debugMsg(String componentName, String dMsg) throws Exception {
    try {

    }
}

PropertyHandler properties = new PropertyHandler("Global");

DEBUG_LOG_LEVEL = properties.getStringValue("audit");
DEBUG_LOG_FILE = properties.getStringValue("debug_log");

writeToFile(componentName, DEBUG_LOG_FILE, dMsg);
}

} catch(InterfacesException ieExc) {
    ieExc.printStackTrace();
}

}

public static void handleAuditLogs(String componentName, String strLogMessage, String strLogLevel) throws Exception {

    try {
    {

        PropertyHandler properties = new PropertyHandler("Global");

        AUDIT_LOG_LEVEL = properties.getStringValue("audit");
        DEBUG_LOG_LEVEL = properties.getStringValue("debug");
        AUDIT_LOG_FILE = properties.getStringValue("audit_log");
        DEBUG_LOG_FILE = properties.getStringValue("debug_log");
        ERROR_LOG_FILE = properties.getStringValue("error_log");

        if (AUDIT_LOG_LEVEL.equals("yes") && strLogLevel.equals("AUDIT")) {
            writeToFile(componentName, AUDIT_LOG_FILE, strLogMessage);
        }
        else if (DEBUG_LOG_LEVEL.equals("yes") && strLogLevel.equals("DEBUG")) {
            writeToFile(componentName, DEBUG_LOG_FILE, strLogMessage);
        }
    }
    
}
public static void handlePerformanceLogs(String componentName, long lStart, String strMsg) throws Exception {

    final String IN SECONDS = " (In Seconds)";
    long lEnd = 0;
    float fTimeInSecs = 0;
    String strTimeTaken = null;

    try {

        PropertyHandler properties = new PropertyHandler("Global");

        PERFORMANCE_LOG_LEVEL = properties.getStringValue("performance");
        PERFORMANCE_LOG_FILE = properties.getStringValue("performance_log");

        if (PERFORMANCE_LOG_LEVEL.equals("yes")) {
            lEnd = System.currentTimeMillis();
            fTimeInSecs = (float)((lEnd - lStart) / 1000.0);
            strTimeTaken = fTimeInSecs + IN SECONDS;

            writeToFile(componentName, PERFORMANCE_LOG_FILE, strMsg+strTimeTaken);
        }
    } catch(InterfacesException ieExc) {
        ieExc.printStackTrace();
    }
}
catch(InterfacesException ieExc)
{
    ieExc.printStackTrace();
}

public static void writeToFile(String componentName, String LOG_FILE, String strLogMsg)
{
    try
    {
        File logFile = new File(LOG_FILE);
        if(!logFile.exists())
        {
            logFile.createNewFile();
        }
        out =new PrintWriter(new BufferedWriter(new FileWriter(LOG_FILE,true)));
        out.println(date("MM-dd-yyyy'@'hh:mm:ss") +TAB+ componentName
                     +TAB+ strLogMsg);
        out.flush();
        out.close();
    }catch(IOException ioe)
    {
        ioe.printStackTrace();
    }
}

public static String date( String format)
{
    SimpleDateFormat formatter = new SimpleDateFormat(format);
}
java.util.Date currentTime = new 
java.util.Date(System.currentTimeMillis());
    String dateString = formatter.format(currentTime);
    return dateString;
}

package com.csus.ctmgmt.common;

public class InterfacesException extends Exception
{
    String strPrefixMsg;
    String strErrorCode;
    String strSuffixMsg;
    Exception wrapException;
    /**
     * Constructor sets the Prefix,ErrorCode and Suffix
     * @param  strPrefix  prefix of error message
     * @param strErrorCode error code
     * @param strSuffix  suffix of error message
     */
    public InterfacesException(Exception wrapException, String strErrorCode,String strSuffix)
    {
        super(wrapException.toString());
        this.wrapException=wrapException;
        this.strErrorCode=strErrorCode;
        this.strSuffixMsg=strSuffix;
    }
    public InterfacesException(String strPrefix,
    String strErrorCode,String strSuffix)
    {
        this.strPrefixMsg=strPrefix;
        this.strErrorCode=strErrorCode;
        this.strSuffixMsg=strSuffix;
    }
    /**
     * Constructor sets ErrorCode and Suffix
     * @param strErrorCode error code
     */
    public InterfacesException(String strPrefix, 
    String strErrorCode,String strSuffix)
    {
*   @param strSuffix suffix of error message
*/
public InterfacesException(String strErrorCode, String strSuffix) {
    this.strErrorCode = strErrorCode;
    this.strSuffixMsg = strSuffix;
    this.strPrefixMsg = "";
}
/**
*   Constructor sets ErrorCode
*   @param strErrorCode error code
*/
public InterfacesException(String strErrorCode) {
    this.strErrorCode = strErrorCode;
    this.strSuffixMsg = "";
    this.strPrefixMsg = "";
}
/**
*   No argument constructor
*/
public InterfacesException() {
    strPrefixMsg = "";
    strErrorCode = "";
    strSuffixMsg = "";
}
/**
*   Getter Methods
*   @return strPrefixMsg prefix message
*/
public String getPrefix() {
    return strPrefixMsg;
}
/**
*   Getter Methods
*   @return strErrorCode error message
*/
public String getErrorCode() {
    return strErrorCode;
}
/**
*   Getter Methods
*   @return strSuffixMsg prefix message
*/
public String getSuffix() {"
public void rethrow() throws Exception {
    throw wrapException;
}

package com.csus.ctmgmt.common;

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

public class DbUtil {

    /**
     * <code> openConnection </code> opens DB connection
     *
     * @param conn SQL Connection object
     * @param strDriverUrl Driver URL
     * @param strUrl Server URL
     * @param strUserName User Name
     * @param strPlainPassword Plain Password
     * @return Connection
     */
    public Connection openConnection(String strDriverUrl, String strUrl, String strUserName, String strPlainPassword) {
        Connection conn = null;
        try {
            Class.forName(strDriverUrl);
            conn = (Connection) DriverManager.getConnection(strUrl, strUserName, strPlainPassword);
            conn.setAutoCommit(true);
        } catch(Exception ex) {
            ex.printStackTrace();
        }
        return conn;
    } // End of method: openConnection

    /**
     * <code> closeConnection </code> closes DB connection.
     * @param conn SQL Connection object
     */
    public void closeConnection(Connection conn) {
        try {
            conn.close();
        } catch(SQLException ex) {
            ex.printStackTrace();
        }
    }

} // End of class: DbUtil
if(conn != null) {
    conn.close();
}
} catch(Exception ex) {
    ex.printStackTrace();
}
} // End of method: closeConnection

/**
 * <code>commit</code> commits database transactions.
 * @param conn SQL Connection object.
 */
public void commit(Connection conn){
    try {
        conn.commit();
    } catch(Exception ex) {
        ex.printStackTrace();
    }
} // End of method: commit

/**
 * <code>rollback</code> rolls back database transactions.
 * @param conn SQL Connection object
 */
public void rollback(Connection conn){
    try {
        conn.rollback();
    } catch(Exception ex) {
        ex.printStackTrace();
    }
} // End of method: rollback

} // end of class: DbUtil
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<%@ page import="java.io.*" %>
<head>
<meta content="en-us" http-equiv="Content-Language" />
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<title>CMS</title>

<link href="../css/style.css" rel="stylesheet" type="text/css" />
<link href="../themes/default.css" rel="stylesheet" type="text/css"/>
<link href="../themes/alphacube.css" rel="stylesheet" type="text/css"/>
<script type="text/javascript" src="../js/jquery.js"></script>
<script type="text/javascript" src="../js/thickbox.js"></script>
<script type="text/javascript" src="../js/bubble.js"></script>
<link rel=stylesheet type=text/css href="../css/thickbox.css">
<script>
function changeBackground(sel) {
    var id2 ;
    id2 = document.getElementById("activenav");
    if( id2 != null) {
        id2.style.backgroundImage="url(<%=out.print(imagesId)%>/L3_nav/bg_bar.png)";
        id2.id = 'notactive';
    }
    sel.id = 'activenav';
    sel.style.backgroundImage="url(<%=out.print(imagesId)%>/L3_nav/selected_cell.png)";
}
</script>

<style>
html {

font-size: 14px;
font-family: Tahoma, Arial, Helvetica, sans-serif;
color: #FF0000;
font-weight: bold;
}

.div-2e {
    position: absolute;
top: 80px;
left: 185px;
width: 60%;
height: 80%;
font-family: Arial, Helvetica, sans-serif;
font-size: 12px;
color: #FFFFFF;
font-weight: bold;
clip: rect(auto, auto, auto, 0px);
}
</style>

<style>
    .div-2atab {
        position: absolute;
top: 5px;
left: 5px;
width: 16;
background-color: #FFEECC;
}

    .div-2btab {
        position: absolute;
top: 5px;
left: 80px;
width: 16;
background-color: #FFEECC;
}

    .div-2ctab {
        position: absolute;
top: 5px;
left: 130px;
width: 16;
background-color: #FFEECC;
Welcome back <%out.print(username);%> !

<a href="color.jsp?TB_iframe=true&height=600&width=800" class="thickbox" style="text-decoration:none;color:#ffffff;">Preferences</a> &nbsp;&nbsp;> &nbsp;&nbsp;About &nbsp;>&nbsp; Help
<tr><td class="style3"><img src="<%out.print(imagesId);%>/line.png"></td></tr><tr><td class="style3"><img src="<%out.print(imagesId);%>/line.png"></td></tr><tr><td class="style3" style="height: 26px;background-image: url(images/gray_bg.png);">Search</td></tr><tr><td class="style3"><img src="<%out.print(imagesId);%>/line.png"></td></tr><tr><td style="height: 26px;" onclick="javascript:changeBackground(this);"><a href="asearch.jsp" target="middlepane"> Advanced Search</a></td></tr><tr><td class="style3"><img src="<%out.print(imagesId);%>/line.png"></td></tr><tr><td style="height: 26px;" onclick="javascript:changeBackground(this);"><a href="picksearch.jsp" target="middlepane"> Pick a Search</a></td></tr>
<tr>
<td class="style3"><img src="<%out.print(imagesId);%>/line.png"/></td>
</tr>
<tr>
<td class="style3"><img src="<%out.print(imagesId);%>/line.png"/></td>
</tr>
<tr>
<td style="height: 26px;" &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>
</tr>
<tr>
<td class="style3"><img src="<%out.print(imagesId);%>/line.png"/></td>
</tr>
<tr>
<td class="style3"><img src="<%out.print(imagesId);%>/line.png"/></td>
</tr>
<tr>
<td style="height: 26px;" &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>
</tr>
<tr>
<td class="style3"><img src="<%out.print(imagesId);%>/line.png"/></td>
</tr>
<tr>
<td class="style3" style="height: 26px;background-image: url(images/gray_bg.png);" &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Administrator</td>
</tr>
<tr>
<td style="height: 26px;" onclick="javascript:changeBackground(this);"&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<a href="http://localhost/grid/term.asp" target="middlepane">Term</a></td>
</tr>
<tr>
    <td class="style3"><img src="&lt;%out.print(imagesId);%>/line.png"></td>
</tr>

<tr>
    <td style="height: 26px" onclick="javascript:changeBackground(this);"><a href="http://localhost/grid/functions.asp" target="middlepane">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Functions</a></td>
</tr>

<tr>
    <td class="style3"><img src="&lt;%out.print(imagesId);%>/line.png"></td>
</tr>

<tr>
    <td style="height: 26px" class="style2" style="height: 26px;background-image:url(images/gray_bg.png);" onclick="javascript:changeBackground(this);"><a href="http://localhost/grid/element.asp" target="middlepane">Subject Elements</a></td>
</tr>

<tr>
    <td class="style3"><img src="&lt;%out.print(imagesId);%>/line.png"></td>
</tr>

<tr>
    <td style="height: 26px" style="height: 26px;background-image:url(images/gray_bg.png);" onclick="javascript:changeBackground(this);">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Reports</td>
</tr>
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create Report</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Select Report</td>
<td></td>
</tr>
</tbody>
</table>

---

<tr>
<td class="style3"><img src="<%=out.print(imagesId);%>/line.png"/></td>
</tr>
<tr style="height: 26px" class="style2" onclick="javascript:changeBackground(this);"><a href="report.jsp" target="middlepane">Create Report</a></td>
</tr>
<tr style="height: 26px" class="style2" onclick="javascript:changeBackground(this);"><a href="report.jsp" target="middlepane">Select Report</a></td>
</tr>
<tr style="height: 26px">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>
</tr>
<tr style="height: 26px">&nbsp;</td>
</tr>
<tr style="height: 26px">&nbsp;</td>
</tr>
<tr style="height: 26px">&nbsp;</td>
</tr>
String optionTerm = "";
String term = request.getParameter("term");
String subid = request.getParameter("subid");
String id = request.getParameter("id");

Connection conn = null;
ConnectToCsus co = new ConnectToCsus();

conn = co.getConnection();
try {

String queryClass = "Select * from csus_student_class where class_id = ? order by id";
PreparedStatement stmtClass = conn.prepareStatement(queryClass);
stmtClass.setString(1, id);
ResultSet rsClass = stmtClass.executeQuery();
String sid = "";
String osid = "";
int score = 0;
int total = 0;
int count = 0;
String insert = "";
int max = 0;
int imax = 0;
String grade = "";

while(rsClass.next()) {

sid = rsClass.getString("student_id");

score = rsClass.getInt("score");
max = rsClass.getInt("max");

total = total + score;

if(sid != null && !sid.equals("") && !sid.equals(osid) && count != 0) {
String delete = "Delete from fct_student where class_id =? and student_id = ?";
PreparedStatement stmtDelete = conn.prepareStatement(delete);
stmtDelete.setString(1,id);
stmtDelete.setString(2,osid);
stmtDelete.executeUpdate();

double gr = ((double)total/(double)imax) ;
out.print(gr+"<br>");
gr = gr * 100;

if(gr > 1 && gr <60)
grade = "D";

if(gr > 60 && gr <70)
grade = "C";

if(gr > 70 && gr <80)
grade = "B";

if(gr > 80)
grade = "A";

insert = "Insert into fct_student (class_id,student_id,score,grade) values(?,?,?,?)";
PreparedStatement stmtInsert = conn.prepareStatement(insert);
stmtInsert.setString(1,id);
stmtInsert.setString(2,osid);
stmtInsert.setInt(3,total);
stmtInsert.setString(4,grade);
stmtInsert.executeUpdate();

total = 0;
imax =0;
}

imax = max + imax;
osid = sid;
count = count + 1;
double gr = ((double)total/(double)imax) ;

    gr = gr * 100;

    if(gr > 1 && gr <60)
        grade = "D";

    if(gr > 60 && gr <70)
        grade = "C";

    if(gr > 70 && gr <80)
        grade = "B";

    if(gr > 80)
        grade = "A";

    String delete1 = "Delete from fct_student where class_id =? and student_id = ?";
    PreparedStatement stmtDelete = conn.prepareStatement(delete1);
    stmtDelete.setString(1,id);
    stmtDelete.setString(2,osid);
    stmtDelete.executeUpdate();

    String insert1 = "Insert into fct_student (class_id,student_id,score,grade)
    values(?,?,?,?)";
    PreparedStatement stmtInsert = conn.prepareStatement(insert1);
    stmtInsert.setString(1,id);
    stmtInsert.setString(2,osid);
    stmtInsert.setInt(3,total);
    stmtInsert.setString(4,grade);
    stmtInsert.executeUpdate();

    conn.close();
    out.print("Grades calculated");
}

catch(Exception ee) {
    out.print(ee);
    try {conn.close(); } catch(Exception eee) {} }
String name = request.getParameter("class");
String subject = request.getParameter("subject");
String faculty = request.getParameter("faculty");
String term = request.getParameter("term");

String operation = request.getParameter("operation");

String optionFaculty = ";"
String optionSubject = ";"
String optionTerm = ";"
String id = "";

Connection conn = null;
ConnectToCsus co = new ConnectToCsus();
conn = co.getConnection();

try {
    String queryFaculty = "Select * from csus_faculty order by name";
    PreparedStatement stmtFaculty = conn.prepareStatement(queryFaculty);
    ResultSet rsFaculty = stmtFaculty.executeQuery();

    while(rsFaculty.next()) {

optionFaculty = optionFaculty + "<option value=" + 
    rsFaculty.getString("faculty_id") + ">
" + rsFaculty.getString("name") + 
"</option>
";
}

String querySubject = "Select * from csus_subject order by subject_name";
    PreparedStatement stmtSubject1 = conn.prepareStatement(querySubject);
    ResultSet rsSubject = stmtSubject1.executeQuery();

    while(rsSubject.next()) {
        optionSubject = optionSubject + "<option value=" + 
    rsSubject.getString("subject_id") + ">
" + rsSubject.getString("subject_name") + 
"</option>
";
    }

String queryTerm = "Select * from csus_term order by name";
    PreparedStatement stmtTerm = conn.prepareStatement(queryTerm);
    ResultSet rsTerm = stmtTerm.executeQuery();

    while(rsTerm.next()) {
        optionTerm = optionTerm + "<option value=" + 
    rsTerm.getString("id") + ">
" + rsTerm.getString("name") + 
"</option>
";
    }

if(operation != null && operation.equals("new")) {

    String insertSubject = "Insert into csus_class (subject_id,term_id,faculty_id,name) 
        values(?,?,?,?)";
    PreparedStatement stmtSubject = conn.prepareStatement(insertSubject);
    stmtSubject.setString(1,subject);
    stmtSubject.setString(2,term);
    stmtSubject.setString(3,faculty);
    stmtSubject.setString(4,name);
    stmtSubject.executeUpdate();
    out.print("The record saved successfully");

    String queryId = "Select max(id) as id from csus_class";
    PreparedStatement stmtId = conn.prepareStatement(queryId);
    ResultSet rsId = stmtId.executeQuery();
while(rsId.next()) {
    id = rsId.getString("id");
}
}
conn.close();
}
catch(Exception ee) {
    out.print(ee);
    try { conn.close(); } catch(Exception eee) {} }

%>

<html>
<head>
<style>
body {
    scrollbar-arrow-color: #464646;
    scrollbar-3dlight-color: #CCCCCC;
    scrollbar-darkshadow-color: #999999;
    scrollbar-face-color: #D4D0C8;
    scrollbar-highlight-color: #FFFFFF;
    scrollbar-shadow-color: #CCCCCC;
    scrollbar-track-color: #e6e6e6;
}
</style>
</head>
<body bgcolor="#b6b7b7">
<div style="background-color:#EFEFEF;position:absolute;top:24px;left:0px;width:100%,height:2100px">
<form name ="f" action = "newclass.jsp?operation=new" method="post">
<table>
<tr>
<td>
<table>
<thead>
<tr>
<th>Class Name:</th>
<th><code>&lt;input type=&quot;text&quot; name=&quot;class&quot; value=&quot;&lt;%out.print(name);%&gt;&quot; style=&quot;width: 232px&quot;&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject Faculty:</td>
<td><code>&lt;select name=&quot;faculty&quot; style=&quot;width: 232px&quot;&gt;</code>&lt;br&gt;<code>&lt;option value=&quot;&quot;&gt;--Select One&lt;/option&gt;</code>&lt;br&gt;<code>&lt;%out.print(optionFaculty);%&gt;</code>&lt;br&gt;<code>&lt;/select&gt;</code></td>
</tr>
<tr>
<td>Subject Term:</td>
<td><code>&lt;select name=&quot;term&quot; style=&quot;width: 232px&quot;&gt;</code>&lt;br&gt;<code>&lt;option value=&quot;&quot;&gt;--Select One&lt;/option&gt;</code>&lt;br&gt;<code>&lt;%out.print(optionTerm);%&gt;</code>&lt;br&gt;<code>&lt;/select&gt;</code></td>
</tr>
<tr>
<td>Subject Subject:</td>
<td><code>&lt;select name=&quot;subject&quot; style=&quot;width: 232px&quot;&gt;</code>&lt;br&gt;<code>&lt;option value=&quot;&quot;&gt;--Select One&lt;/option&gt;</code>&lt;br&gt;<code>&lt;%out.print(optionSubject);%&gt;</code>&lt;br&gt;<code>&lt;/select&gt;</code></td>
</tr>
</tbody>
</table>
<td>
<input type="submit" name="Submit" value="Submit"/>
</td>
</tr>
</form>
</table>
<table>
<tr>
<% if (operation != null && operation.equals("new")) { %>
<iframe src ="http://localhost/grid/student.asp?sub=<%out.print(id);%>" name="cseu" width="1000px" height="1000px"></iframe>
<% } %>
</tr>
<script>
var optionst = document.f.term.options;
for (var i = 0; i < optionst.length; i++) {
    if (optionst[i].value == '<%out.print(term);%>') {
        optionst[i].selected = true;
    }
}
</script>
<script>
var optionst = document.f.faculty.options;
for (var i = 0; i < optionst.length; i++) {
    if (optionst[i].value == '<%out.print(faculty);%>') {
        optionst[i].selected = true;
    }
}
</script>
<script>
var optionst = document.f.subject.options;
for (var i = 0; i < optionst.length; i++) {
    if (optionst[i].value == '<%out.print(subject);%>') {
        optionst[i].selected = true;
    }
}
</script>
for (var i = 0; i < optionst.length; i++) {
    if (optionst[i].value == '<%out.print(subject);%>') {
        optionst[i].selected = true;
    }
}
</script>
</table>
</html>
BIBLIOGRAPHY


[4]  A webpage which explains System Reference Cursor support for Oracle 8i and above –
   http://www.developerfusion.co.uk/show/671/12/

   http://www.developerfusion.co.uk/show/671/11/


   Data Warehousing

[8]  http://edw.berkeley.edu/SDW_data_dictionary.htm  Student datamart example