**Animon.sys Enrollment System**

Logical and Physical Data Models and Queries

for the course on

Introduction to Databases

(INTRODB)

Submitted by:

Aquino, Kurt Neil

Choy, Matthew Seaver

Hade, Alden Luc

Ms. Ethel Ong

Teacher

August 06, 2014

1. Relational Data Model

Department

|  |  |  |
| --- | --- | --- |
| dept\_num | dept\_name | dept\_head |

Faculty

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| faculty\_id | last\_name | first\_name | email\_add | hire\_date | to\_date | dept\_num |

Course

|  |  |  |  |
| --- | --- | --- | --- |
| course\_code | course\_name | units | dept\_num |

Offering

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| course\_num | term | acad\_year | section | sched\_days | start\_time |
| end\_time | capacity | enrolled | course\_code | faculty\_id |  |

Transaction Record

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| transaction\_id | date\_stamp | time\_stamp | transaction\_type | student\_id | course\_num |

Student

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| student\_id | last\_name | first\_name | email\_add | degree |
| enroll\_date | to\_date | status | cgpa | term\_units |

1. Queries that provides access to the database

Student’s Features:

1. SELECT \* FROM offering;

* This is a query statement made to retrieve all the current offerings in the term.

2. SELECT \* FROM course;

* This is a query statement made to retrieve all (distict) the current courses being offered.

3. SELECT \* FROM offering O, faculty F WHERE O.faculty\_id = F.faculty\_id;

* This is a query statement made to check the offerings each professor handles.

4. SELECT \* FROM offering O WHERE O.course\_code = userInput;

* This is a query statement made to find a specific course offering using its course code based on user input (userInput = nextLine()).

5. SELECT \* FROM offering O WHERE O.course\_num = userInput;

* This is a query statement made to find a specific course offering using its course number based on user input (userInput = nextInt()).

6. SELECT \* FROM offering O, faculty F WHERE O.faculty\_id = F.faculty\_id AND F.last\_name = userInput;

* This is a query statement used to list the offerings of a certain professor based on his/her last name. (userInput = nextLine())

7. SELECT \* FROM offering O WHERE (O.capacity - O.enrolled) > 0 AND O.course\_code = userInput;

* This is a query statement for finding offerings with at least one slot left based on its course code. userInput = (nextLine())

Faculty’s Features:

1. SELECT \* FROM offering O, faculty F WHERE O.faculty\_id = F.faculty\_id;

* This is a query that displays the current offerings assigned to the professor for the term. The faculty id is based on the log-in id of the user itself.

1. SELECT \* FROM Student S, transaction\_record TR, faculty F, Offering O WHERE s.student\_id = TR.student.id AND TR.course\_num = O.course\_num AND O.faculty\_id = F.Faculty\_id AND TR.transaction\_type = “ADD”;

* This is a query for finding the entire list of students enrolled under a specific professor based on his/her faculty\_id.

1. SELECT O.enrolled FROM offering O, faculty F WHERE O.faculty\_id = F.faculty\_id AND O.course\_num = userInput;

* This is a query for finding how many students has enrolled to an offering of a certain professor. The faculty\_id is based on the user login and the course number is based on user input. (userInput = nextInt())

Administrator’s Features:

1. SELECT \* FROM student;

* This is a query statement made to retrieve all the current students in the database.

2. SELECT \* FROM transaction\_record;

* This is a query statement made to retrieve all the transactions that have been done.

3. SELECT \* FROM faculty;

* This is a query statement made to retrieve all the current faculties in the database.

4. SELECT \* FROM department;

* This is a query statement made to retrieve the current departments in the database.

5. SELECT \* FROM transaction\_record TR

WHERE TR.student\_id = userInput;

* This is a query statement made to retrieve the current transactions a particular student has made based on user input. (userInput = nextInt())

6. SELECT \* FROM student S, transaction\_record TR

WHERE TR.student\_id = S.student\_id and transaction\_type = “ADD”;

* This is a query statement made to retrieve the added classes of students.

7. INSERT INTO transaction\_record(transaction\_id, date\_stamp, transaction\_type, student\_id, course\_name)

VALUES (trans\_no, 'date(now)', 'ADD’, '11313803', 'INTRODB');

* This is a query statement made to insert new transactions.

8. INSERT INTO offering(course\_num, term, acad\_year, section, sched\_days, start\_time, end\_time, room\_num, capacity, enrolled, course\_code, faculty\_id)

VALUES('488941', '3', '2014-2015', 'S17', 'MW', '0915- 1045', 'G306'. '45', '0', 'INTRODB', '1001');

* This is a query statement made to insert new course offerings. The actual values to be input is acquired through the application itself and then stored into the database.

3. Query Statements Used for Each Report

1. **Total Units Each Student Has:**

SELECT first\_name, last\_name, O.course\_code, sum(distinct C.Units)

FROM Offering O, Course C, transaction\_record TR, student S

WHERE O.course\_code = C.course\_code and O.course\_num = TR.course\_num and TR.student\_id = S.student\_ID and transaction\_type = 'ADD'

GROUP BY S.student\_id;

* This statement generates the number of units each student has currently enrolled in.

1. **Number of Transactions today:**

SELECT count(\*), date(now())

FROM transaction\_record TR

WHERE date(now()) = date\_stamp;

* This statement generates the number of transactions today.

1. **Transactions of today:**

SELECT \*

FROM transaction\_record

WHERE date\_stamp = date(now());

* This statement generates all the transactions that happened today