# *Database Management II (420-D20-HR)*

# *Lab 9 –* CRUD Matrix and Optimization

Date assigned: Tuesday, April 11, 2017

Date due: Tuesday, April 11, 2017 16:50

**Objectives:**

At the end of this lab you will be able to:

1. map data attributes to processes using a CRUD matrix;
2. evaluate the advantages and disadvantages of denormalizing a data model to improve performance.

**To Start:**

1. Rename this document to ***username*\_D20\_L09\_Optimization.docx**.

**To Be Handed In:**

1. Your ***username*\_D20\_L09\_Optimization.docx** should be uploaded to Moodle.

# CRUD Matrix

Consider the following processes for the Indo-US database:

1. Print the report cards for a term. The report card should include the **term start** and **end dates**, the **student’s name**, **number** and **address** and the course number, name, credits and mark for each course taken during the term. The grade point average should also be output for the student[[1]](#footnote-1). **The user will be prompted for the term id.** (*Report Cards)*
2. Print the schedule for a faculty member given the faculty name. The schedule includes the course number and name, the day and start and end times for the course for each course taught by the faculty member in the current semester. (*Faculty Timetable*)
3. Update all the final marks for a course given the teacher name, course number, section number and termid. (*Update Final Marks*)
4. Register an existing student for a course. (*Registration*) The student number and course id will be entered. Registration involves the follow process:
5. If the student exists
   1. While the student is not registered and there is another section for the course
      1. If there is room in the section
         1. If the student is not registered in another course at the same time
            1. Add a new row to the REGISTRATION table for the student.

***To Do:***

## Complete the CRUD matrix for the Indo-US database in ***username\_*D20\_L09\_Optimization\_Answers.docx** for the four processes described above.

| **Table . Attribute** | * 1. **Process Report Cards** | * 1. **Faculty Timetable** | * 1. **Update final marks** | * 1. **Registration** |
| --- | --- | --- | --- | --- |
| **IU\_MAJOR** |  |  |  |  |
| **.MajorId [PK]** |  |  |  |  |
| **.MajorDesc** |  |  |  |  |
| **IU\_STUDENT** |  |  |  |  |
| **.StudentId [PK]** | sR(5) |  |  | dR(1) |
| **.Last** | r |  |  |  |
| **.First** | r |  |  |  |
| **.Street** | r |  |  |  |
| **.City** | r |  |  |  |
| **.Province** | r |  |  |  |
| **.PostCode** | r |  |  |  |
| **.StartTerm** |  |  |  |  |
| **.BirthDate** | r |  |  |  |
| **.Phone** |  |  |  |  |
| **.MajorId [FK]** |  |  |  |  |
| **.FacultyId [FK]** |  |  |  |  |
| **IU\_FACULTY** |  |  |  |  |
| **.FacultyId [PK]** |  | R | r |  |
| **.Name** |  | sR(1) | sR(1) |  |
| **.Phone** |  |  |  |  |
| **.DeptId [FK]** |  |  |  |  |
| **.RoomId [FK]** |  |  |  |  |
| **IU\_DEPARTMENT** |  |  |  |  |
| **.DeptId [PK]** |  |  |  |  |
| **.DeptName** |  |  |  |  |
| **.FacultyId [FK]** |  |  |  |  |
| **IU\_REGISTRATION** |  |  |  |  |
| **.Csid [PK][FK]** | sR(3) |  | sR(3) |  |
| **.StudentId [PK][FK]** |  |  |  |  |
| **.MidTerm** |  |  |  |  |
| **.Final** |  |  | U |  |
| **.RegStatus** |  |  |  |  |
| **IU\_CRSSECTION** |  |  |  |  |
| **.Csid [PK]** |  | r |  |  |
| **.Section** |  |  | sR(2) |  |
| **.Day** |  | r |  |  |
| **.StartTime** |  | r |  |  |
| **.EndTime** |  | r |  |  |
| **.MaxCount** |  |  |  |  |
| **.Courseid [FK]** |  |  | sR(2) |  |
| **.TermId [FK]** | sR(2) | sR(3) |  |  |
| **.RoomId [FK]** |  |  |  |  |
| **.FacultyId [FK]** |  | sR(3) | sR(2) |  |
| **IU\_COURSE** |  |  |  |  |
| **.Courseid [PK]** | sR(4) | sR(4) |  | dR(1) |
| **.Title** | r | r |  |  |
| **.Credits** | r |  |  |  |
| **IU\_LOCATION** |  |  |  |  |
| **.Roomid [PK]** |  |  |  |  |
| **.Building** |  |  |  |  |
| **.RoomNo** |  |  |  |  |
| **.Capacity** |  |  |  |  |
| **.RoomType [FK]** |  |  |  |  |
| **IU\_TERM** |  |  |  |  |
| **.Termid [PK]** | dR(1) | r |  |  |
| **.TermDesc** |  |  |  |  |
| **.StartDate** | r | sR(2) |  |  |
| **.EndDate** | r | sR(2) |  |  |
| **IU\_ROOM** |  |  |  |  |
| **.RoomType [PK]** |  |  |  |  |
| **.RoomDesc** |  |  |  |  |
| **Totals** |  |  |  |  |
| **.Number of Entities Read** | **5** | **4** | **3** | **3** |
| **.Number of Entities Created** | **0** | **0** | **0** | **0/1** |
| **.Number of Entities Updated** | **0** | **0** | **1** | **0** |
| **.Number of Entities Deleted** | **0** | **0** | **0** | **0** |

# Optimization

## Decide whether or not name should be an alternate key to the faculty table based on the CRUD matrix. Identify the process(es) that would use it for access.

Yes, people will search for Faculty members by name, so it should be an alt name

However, these are probably things you won’t use that much and are easy to pull up, so why even really bother?

## Determine the best order for the primary key of the IU\_REGISTRATION table. Justify your answer based on the information in the CRUD matrix.

**Order of Keys**: CSID, STUDENTID

**Rationale**: The IU\_REGISTRATION table is accessed more through the CSID than the StudentId and I think that would be a determining factor.

## Complete the analysis of each of the following optimizations based on your CRUD matrix.

#### Define a supplementary relationship between IU\_FACULTY and IU\_CRSSECTION for courses taught in the current semester.

***How it improves performance:***

*I means that you don’t need to do a sequential read through the crssection table to find the courses taught by the faculty member*

***Costs:***

*You have a bunch of extra data inside the CRSSECTION table, which clutters things up and gives a lot of repeat data*

***Your recommendation:***🞏 I recommend this optimization

🞏 I do not recommend this optimization

***Justification:***

***Too much maintenance and repetitive data***

#### Repeat the COURSE title in the IU\_CRSSECTION table.

***How it improves performance:***

*Adding the course title to the CRSSECTION table makes things a lot easier since you don’t need to go and do a sequential read through the course table, then do another through the crssection table. You’re likely going to search by human readable content the most, so it takes a lot less data to search through*

***Costs:***

*You have one extra column in the CRSSECTION table to keep up to date, but will likely rarely need to be updated anyways*

***Your recommendation:***🞏 I recommend this optimization

🞏 I do not recommend this optimization

***Justification:***

***Little extra maintenance and data while making a lot of queries far more efficient***

#### Add grade point average to the IU\_STUDENT table.

***How it improves performance:***

*The calculation doesn’t need to be done every time you want to access a simple and important piece of data.*

***Costs:***

*A simple conversion calculation*

***Your recommendation:***🞏 I recommend this optimization

🞏 I do not recommend this optimization

***Justification:***

*The calculation would need to be done every time you want to access that student’s GPA, which means a search through the REGISTRATION table and the COURSE table for credits, which could be avoided by just doing it once and grabbing it every time. It drops the O(N) drastically every time you want access the data.*

#### Subdivide the IU\_REGISTRATION table into a CURRENT REGISTRATION table and a COMPLETED COURSE table.

***How it improves performance:***

*You don’t have to determine whether the course is active or not, and you don’t need to include any logic for determining whether or not the course can be updated or something. You aren’t ever going to be updating a completed course for a student (unless under special circumstances) and dividing it between two tables makes that access control easier to implement and maintain.*

***Costs:***

*You have similar pieces of data separated between two different places. Every time someone completes a course, which is going to be several at a time, you have to move the data from one table to another, then remove it from the current table. It’s a lot of extra work instead of just having a flag.*

***Your recommendation:***🞏 I recommend this optimization

🞏 I do not recommend this optimization

***Justification:***

*It’s a lot of extra work and maintenance when you can just have a flag.*

# Assessment

1. What did you learn in completing this lab?

CRUD and some interesting optimization things that I’d never really thought about. I also learned a lot about how querying languages work in the backend, because the logic of looping is so hidden from us that in a 5 table query, we don’t see how inefficient that is, and it how my denormalizing it, we can make the DB much more efficient.

1. What did you have difficulty with?

CRUD table

1. What did you do well?

I think I did most of it

1. How many hours did you spend in completing this lab?

2

1. What took you the most time?

1. To calculate the grade point average, each letter grade is converted to a numeric value and the average of the numbers is calculated. (The grade equivalencies are: A:4, A-: 3.7, B+:3.3, B:3, B-:2.7, C+:2.3, C:2, C-:1.7, D+:1.3, D:1, D-:0.7, F:0) [↑](#footnote-ref-1)