Project 5 CS60 Fall 2017

Designing normalized tables—Part 2 of 2 with 2nd and 3rd Normal Form Faults

Due Monday October 30 at 10:30pm for Section 4052. Due Tuesday October 31 at 10:30pm for Section 4127

This project continues the tasks begun with Project 4 to detect and correct normalization faults. You need to recognize and correct 2nd and 3rd normal form faults in table designs, which lead to redundancies. The project also gives you experience using composite (multi-column) primary and foreign keys.

Unlike Santa Monica College that uses a 4-digit number to identify a section of a course, the tables use two columns (Department\_Name and Course\_Number) to identify a course, and uses three columns (Department\_Name, Course\_Number, and Section\_Letter) to identify a section of a course. Examples:

**Computer Science 101 A** Two sections of the course **Computer Science 101**

**Computer Science 101 B**

**Computer Science 201 A** Three sections of the course **Computer Science 201**

**Computer Science 201 B**

**Computer Science 201 C**

**Biology 101 A** Two sections of the course **Biology 101**

**Biology 101 B**

Please use this method to identify courses and sections in this project; don’t create new surrogate keys for courses and sections. Similarly, in this project, identify a classroom or other room by the combination Building\_Name + Room\_Number, such as Business 253

Name the tables and columns according to Oracle’s rules:

● Names begin with a letter

● Names can include letters, decimal digits as characters, underscore (\_), pound sign (#), and dollar sign ($)

No spaces, periods, hyphens, and other characters except those listed above

● Maximum of 30 characters

● Names are descriptive

Alter the existing tables and create any new tables with their table names, column names, primary keys, and any foreign keys so the tables satisfy the third rule of normalization. Hence, each table satisfies 1NF, 2NF, and 3NF. In each table, indicate the primary key with ***PK*** after the column(s), and any foreign key(s)with ***FK*** after the column(s). Using the dot notation, *Tablename.Columnname*, identify the table and columns (the PK) that the foreign keys reference. If more than one foreign key appears in any table, number the keys (e.g., ***FK1***).

To save typing time, use the same format as I have on the following page. Copy from AcShare the file **CS60 Project 5 2017 Fall.docx.** Rename the file and **type in the footer the name of your file: CS60\_5\_*SectionNumber*\_*LastName*\_*FirstName*.doc** where *SectionNumber is your section number (4052 meets Mondays, 4127 meets Tuesdays), and LastName* and *FirstName* are your names. Then edit the table. You can right-click on a row and select ***Insert*** (row) to add a new blank row. Select a row, right-click, and select ***Delete*** (row) to delete a row. Delete any comments I’ve added about the faults, the notes at the bottom, and this first page.

Copy and paste your file into the CS60 folder on server **Zeus**.

|  |  |  |  |
| --- | --- | --- | --- |
| **TABLE NAME** | COLUMN NAME Table and column names must satisfy Oracle’s naming rules listed on the first page--rules that include no spaces or hyphens. | **KEYS** | For a foreign key, the table and column(s) it references |
|  |  |  |  |
| Student | Student ID | PK |  |
|  | Student Lastname |  |  |
|  | Student Firstname |  |  |
|  | Student Middlename |  |  |
|  |  |  |  |
| Enrollment | Department Name | PK |  |
|  | Course Number | PK |  |
|  | Section Letter | PK |  |
|  | Student ID | FK | Enrollment.Student\_ID refs Student.Student\_ID |
|  | Department Chair ID (2NF violation since it depends only on the Department name, which is only part of the primary key in this table.) |  |  |
|  | Student Major (2NF violation; See Note 1 below) |  |  |
|  | Number of Students in Course (2NF violation Note 2) |  |  |
|  | Professor ID (Note 3) (2NF violation because it depends only on the Department Name, Course Number, and Section Letter, which are only part of the PK) |  |  |
|  | Professor Lastname (3NF violation) |  |  |
|  | Professor FIRST NAME (3NF violation) |  |  |
|  | Student Birthdate (2NF violation) |  |  |
|  | Professor Birthdate (3NF violation) |  |  |
|  |  |  |  |
| Course | Department Name | PK |  |
|  | Course Number | PK |  |
|  | Department MAIN OFFICE ROOM (2NF violation) |  |  |
|  | Department MAIN OFFICE BUILDING (2NF violation) |  |  |
|  |  |  |  |
| Section | Department Name | PK, FK |  |
|  | Course Number | PK, FK |  |
|  | Section Letter | PK |  |
|  | Building Name (Note 4) |  |  |
|  | Room Number (Note 4) |  |  |
|  | Seats in Room (3NF violation because it depends only on Building Name and Room Number) |  |  |
|  | Area of Building (square footage; 3NF violation because it only depends upon Building\_Name) |  |  |

Note 1: This is the Department\_Name for the major of the student with ID being Student ID. Assume that students have only one major each.

Note 2. This is the total number of students enrolled in **all sections** of this course. Is this an attribute of ENROLLMENT or SECTION or COURSE?

Note 3. This is the ID of the professor who’s teaching the section that the student is enrolled in. You need to store somewhere who’s teaching every section.

Note 4. Building name and room number store data like Business 255. Different buildings can have the same room, but the combination of Building\_Name and Room Number is unique.