Case Study: Redwood University

A university has requested the design and implementation of a database to store its data. The university encompasses multiple departments, each of which has a chair. The university does not want to store particular information regarding the chair, rather information pertaining to the department name and chair name, as well as the number of faculty members the department has. Department names must always start with Department.

The university has numerous students and each of them has declared at least one major. Additionally, the name and initials of a student are stored. Initials must be more than one character long. For each major, the university wants to store the major name, the department it is associated with, and a code. For example, ‘Biology’ is associated with department 3 (i.e., the Department of Biology) and has the code ‘BIO’. Major codes must be three characters. Majors can be declared by one or more students. A major references one department, however a department offers one or more majors.

Each department has the possibility of hosting events, and an event can be (collaboratively) hosted by one or more departments. In addition to the event name, the university would like to store the start and end dates of the event. As it is logical, an event cannot end before the start date. Information pertaining to events are stored ahead of time, therefore at the time of insertion an event cannot be a past date or the current date. Students must attend one or more events, and each event will comprise one or more students.

图示

中度可信度描述已自动生成2. Develop a logical data model based on the following requirements:

a. Derive relations from the conceptual model.  
Red: Primary Key

Blue: Foreign Key

b. Validate the logical model using normalization to 3NF.

1NF:

Department: Pass  
Major: Pass

Event: An Event Could host by multiple departments and joined by multiple students, so not a pass. Create new table to deal with this problem.

Student: A student could have not only one major and a major could have many students, so not a pass. Create new table to deal with this problem.

图形用户界面, 图示

描述已自动生成After 1NF:

2NF:

Department: Only one primary Key, Pass

Major: Only one primary Key, Pass

Event: Only one primary key, Pass  
 Student: Only one primary key, Pass

Department\_Event: No non-primary key, Pass

Student\_Event: No non-primary key, Pass

Major\_Student: No non-primary key, Pass

After 2NF:

图形用户界面, 图示

描述已自动生成3NF:

Department: All other keys depends on dep\_id, Pass

Major: All other keys depends on major\_Code, Pass

Event: All other keys depends on event\_id, Pass  
 Student: stu\_Initial is depends on fName+lName, so not a pass, but make them a separate table will add a lot more work and might cause multi to multi problem hence adds a lot of unnecessary calculation, so consider keep it, although it is not meet the 3NF standard.

Department\_Event: No non-primary key, Pass

Student\_Event: No non-primary key, Pass

Major\_Student: No non-primary key, Pass

After 3NF:   
图形用户界面, 图示

描述已自动生成 c. Validate the logical model against user transactions.

Q1: List all the student whose major is computer science

A: First use select method in Major Table, find the Major’s id, then select all the stu\_id with that major\_Code in Major\_Student table. If you want the detail of student, union Student table with stu\_id as the foreign key.

Q2: List all Events that’s ends after a select date

A: Select \* form Event table, with a constrain that limits event\_endDate > select date.

Q3: Find all deprtments with over 10 pops, sorting them by pop number high to low.

A: Select \* from Department table, with a constrain that limits dep\_stuffPop > 10, Order by dep\_stuffPop DESC.

Q4: Insert a new student with major “Computer Science” and “Criminology”

A: Find major\_Code in Table “Major” with constrain Having “%Computer Science%” or “%Criminology%”, insert student information into Student Table and insert searched major\_Code and stu\_id into the Major\_Student Table.

Q5: Find all events a student has attended with student’s id.

A: Select \* in Event table with Event table Union Student\_Event Table with event\_id imported from Event Table as foreign key in Student\_Event Table. Where stu\_id = given id.

d. Define integrity constraints:

i. Primary key constraints.

dep\_id, major\_Code, event\_id, stu\_id, (dep\_id, event\_id), (major\_Code, stu\_id), (stu\_id, event\_id) must be unique and not null.

ii. Referential integrity/Foreign key constraints.

dep\_id in Major, (dep\_id, event\_id) in Department\_Event, (major\_Code, stu\_id) in Major\_Student, (stu\_id, event\_id) in Student\_Event must be not null and has the same value in the table it originally come from. When deleting/modifying, they must be change/delete same time with the data they originally come from.

iii. Alternate key constraints (if any).

Dep\_Name in Department, major\_Name in Major must be unique and not null.

iv. General constraints (if any).

All code and ids must be int type, when inserting, event\_startDate and event\_endDate must be later or equal than today. event\_endDate must be later or equal than event\_startDate. Stu\_Initial must equal to the first litter of stu\_fName + stu\_lName. All entrees must be not null except chair\_fName, chair\_lName, dep\_stuffPop. chair\_fName and chair\_lName (if its not null) must be same with one of stu\_fName and stu\_lName in Student Table.

e. Generate the E-R diagram for the logical level (contains FKs as attributes). 图示

描述已自动生成