Constraints Explained

CSC 540 - Project 1 - Spring 2017

Constraints in ER Model

Key Constraints (Primary/Foreign Key)

1. Hierarchical inheritance between *Faculty*, *Admin*, and *Students* to *UnityUsers*. **UnityID** is the primary key for *UnityUsers*, and it travels down as a foreign key to above three tables.
2. *Students* and *Faculty* are associated with *Department* based on foreign key **DeptID**.
3. Particular *CreditRequirements* are mapped to *Students* by foreign key **CreditReqID**.
4. *Courses* have a foreign key from *Department*, **DeptID**, which in combination with **CourseID**, acts as the primary key for the table.
5. *Semesters* are identified by their primary key **SemesterID**, which is auto-incremented based on combinations of (**Season**-**Year**).
6. A unique *CourseOffering* is identified based on *CourseOfferingID*, which is auto-incremented for unique combinations of **CourseID**, **DeptID**, **SectionID**, and **SemesterID**.
7. Table of *Billing* has a foreign key **UnityID** from *Students*, and possesses a weak relationship.
8. *Transcript* and *Waitlist* both have foreign keys - **UnityID** from *Students* and **CourseOfferingID** from *CourseOffering**. Their combination acts as a primary key for these tables.
9. *PermissionMapping* handles unique combinations of *SpecialPermissions* and *CourseOfferings* based on foreign keys (**SpecialPermissionID**, **CourseOfferingID**).
10. Facilitates has a pairing of **UnityID** from *Faculty* and **CourseOfferingID** from *CourseOffering*.
11. As *BillingRate* is identified per *Credit Requirement* per *Semester*, it offers a foreign key matching of **CreditReqID** and **SemesterID**.
12. *CourseOfferings* have pre-requisites of *Courses* mentioned in *Prerequisites* relationship table by **CourseOfferingID** - (**CourseID**, **DeptID**) combination.
13. *Students* can submit *SpecialPermission* requests, which are registered by keys **UnityID** (*Students*), **CourseOfferingID** (*CourseOffering*). And are approved by **UnityID** (*Admin*).

Uniqueness Constraints

- *Students*, *Admins*, and *Faculties* can be uniquely identified based on their corresponding member **ID card numbers**, apart from *UnityID*s.
- *Employees* (*Admin*, *Faculty*) can also be uniquely identified by **ssn**.
- *CourseOfferings* are uniquely listed by **CourseOfferingID**, *Semesters* by **SemesterID**, and *CreditRequirements* by **CreditReqID**.

Data Entry Constraints

There are certain fields in database which encode information in numbers. They have been encoded by Data Entry Constraints as follows - * **residencyLevel** for *Students* from *CreditRequirements* = (1-Undergraduate, 2-Graduate). * **participationLevel** for *Students* from *CreditRequirements* = (1-InState, 2-OutOfState, 3-International). * **Grades** have to be between the possible grade range of [A+, C-]. * **ClassDays** are numbers between (0-9). 0:DE, 1-5:M-F, 6:MW, 7:TTH, 8:MWF, 9:Others

Then there are certain inherent and basic data entry constraints, such as - * *GPA* has to be between

(0, 4.333). * *LastDropDay* can't be before *LastEnrollmentDay* for a semester. Same goes for *Start* and *End* days. * Students can't take a course for less than 1 credit hours (transcript). * For any given CourseOffering, number of available seats cannot be greater than totalSeats offered for that offering.

All of the above constraints have been handled in SQL code in database. But the mapping for encoded elements has been handled at the application level. Database just handles the numbers. The string representations for those (except Grades) are done in Java code.

Constraints in Application Code