Constraints for DB Assignment

Primary Keys:

* Student Table: studentID
* Lecturer Table: lecturerID
* StudentRegistration Table: studentID; this is because a Student can’t be registered twice within a university; this would cause data redundancy. In the scenario of changing course, the table should updated not ignored hence the use of StudentID as primary key.
* --RE THINK StudentContact Table: studentID; in this scenario there will be a one to one mapping of student to studentContact; the university will provide one email to the student so they will only appear once within this table.
* NextOfKinContact Table: studentID. Technically speaking, a student can either have no or one next of kin. This means that each student will appear no more than once in the table and so studentID will be a unique identifier in the table. If they are absent of one, they will either not be in the table or NULL values may be set.
* LecturerContact Table: lecturerID. Within a university, a lecturer will be assigned a single university email and they will have at most one office and so lecturerID can be considered a unique identifier in the table.
* Tutor: studentID. A student can only have one personal tutor within the university and so each student may only appear once within the table.
* Titles: titleID.
* Registration Type: registrationTypeID.

Foreign Key Relationships:

* Student Table: titleID references the titleID attribute in Titles.
* Lecturer Table: titleID references the titleID attribute in Titles.
* StudentRegistration Table: studentID references studentID in Student, registrationTypeID references registrationTypeID in RegistrationType.
* StudentContact Table: studentID references studentID in Student.
* NextOfKinContact Table: studentID references studentID in Student.
* LecturerContact Table: lecturerID references lecturerID in Lecturer.
* Tutor Table: studentID references studentID in Student, lecturerID references lecturerID in Lecturer.

DELETE constraints:

* Any attribute with a foreign key relationship with titleID from Titles should use the ON DELETE RESTRICT constraint as titles are fixed as far as their names go and a title will never be needed to be deleted and so it would make no sense to allow this and so it is restricted.
* Any attribute with a foreign key relationship with registrationTypeID from RegistrationType should use ON DELETE RESTRICT; we don’t want to delete these as they are standard and are concrete information. As with titles, there is no logical reason for wanting to delete them; deleting one with say CASCADE for example would make no sense so I believe this to be the most appropriate constraint.
* Anything with a foreign key relationship with studentID should use ON DELETE CASCADE as if a student is deleted from the database, the other information specified in these tables becomes defunct and useless and so should also be deleted so as to not waste space in the database.
* Anything with a foreign key relationship with lecturerID should use ON DELETE CASCADE for essentially the same reasoning behind using it with studentID, lecturer office information needs to be deleted if the lecturer leaves as it is redundant and it would also cause issues with other constraints if another lecturer takes the office and the old information is still here (this a constraint explained earlier).

UPDATE constraints:

* Any attribute with titleID from Titles as a foreign key relationship should use ON UPDATE RESTRICT. This means we are restricting updates on the Titles table. This makes sense because if you change a title from Mr to Mrs it will cause a lot of incorrect information in the database which we really don’t want. In the scenario of wanting to change title, a new one should be changed and then student/lecturer titles updated where necessary.
* Any attribute using RegistrationTypeID from RegistrationType as a foreign key relationship should use ON UPDATE RESTRICT. This is for the same reasons as with Titles. Registration type normal for example, will never become defunct and so there will not be a scenario to change every normal registration to something for else for example. Changing it otherwise could change all rows using it, potentially causing incorrect data to be shown.
* Anything using studentID or lecturerID from Student and Lecturer appropriately as a foreign key relationship should use ON UPDATE CASCADE. This is because in either case, if we updated the id of a student/lecturer we need to appropriately change all rows in tables representing the same person by adjusting the id appropriately. If we didn’t do this we could end up ruining the integrity of our database.

Type constraints:

* Student Table:
  + studentID: INTEGER
  + titleID: INTEGER
  + foreName: CHAR(40)
  + familyName: CHAR(40)
  + dateOfBirth: DATE
* Lecturer Table:
  + lecturerID: INTEGER
  + titleID: INTEGER
  + foreName: CHAR(40)
  + familyName: CHAR(40)
* StudentRegistration Table:
  + studentID: INTEGER
  + yearOfStudy: INTEGER
  + registrationTypeID: INTEGER
* StudentContact Table:
  + studentID: INTEGER
  + name: CHAR(80)
  + eMailAddress: CHAR(254)
  + postalAdress: CHAR(300)
* NextOfKinContact Table:
  + studentID: INTEGER
  + name: CHAR(80)
  + eMailAddress: CHAR(254) – maximum length email possible
  + postalAddress: CHAR(300)
* LecturerContact Table:
  + lecturerID: INTEGER
  + office: CHAR(7) –may include letters
  + eMailAddress: CHAR(254)
* Tutor Table:
  + studentID: INTEGER
  + lecturerID: INTEGER
* Titles Table:
  + titleID: INTEGER
  + titleString: CHAR(10)
* RegistrationType Table:
  + registrationTypeID: INTEGER
  + description: CHAR(30)

Other constraints:

* dateOfBirth should be > 1900-01-01 and < NOW
* emailAddress must be in format ‘\_%@\_%’
* titleString in Titles must be UNIQUE and NOT NULL
* yearOfStudy in StudentRegistration must be BETWEEN 1 AND 5
* description in Registration should be UNIQUE, NOT NULL to avoid redundancy
* lecturerID in Tutor should be NOT NULL because the table should only show valid combinations of the two; a student with no tutor shouldn’t be in the table.
* foreName and familyName should not contain the space character as that isn’t valid for names split in two.
* forename NOT NULL (you need a first name but you don’t necessarily need a surname so only foreName NOT NULL)
* When considering a table that can have eMailAddress OR postalAddress, you only need one to be contactable so there is a constraint that one can be null but the other must not be null in that case. Both can be not null though.
* Although some people won’t have next of kin for various reasons (and name and all other fields will be null), the people that do (i.e specify a name) must specify one of the two of the aforementioned email or postal address for the same reasons as previously mentioned.
* In lecturerContact the lecturer must have one of an office or an email address (i.e one can be null but not both) in order to make it possible to contact them; there may be an odd scenario where a lecturer doesn’t have an office and just travels to the University for lectures for example.
* The office field in lecturerContact must be UNIQUE. Lecturers don’t share offices.
* All email Addresses must be UNIQUE; no two people can have the same email address.
* The yearOfStudy must be NOT NULL. You cannot register at a university without knowing the year of entry of course, hence the constraint.
* The registrationTypeID in StudentRegistration must be NOT NULL. Again, you need to know how you are registering in order to register in the first place, hence this field is considered mandatory.