COMP3311 20T3 Database Systems

Assignment 2
The MyMyUNSW Database

Last updated: **Sunday 25th October 1:33am**
Most recent changes are shown in red ... older changes are shown in brown.

[Assignment Spec] **[Database Design]** [Schema in SQL] [check1.sql] [Examples] [Fixes+Updates]

Introduction

This document contains a description of the data model and SQL schema for the MyMyUNSW database.

Background

UNSW handles its administrative information using a version of the Peoplesoft product called Campus Solutions. This system is normally accessed via the MyUNSW portal and is maintained by the NSS unit in UNSW IT, and so it's variously called "PeopleSoft", "Campus Solutions", "MyUNSW" and "NSS". The database behind the system is hosted on a large Oracle server, and throughout this document will be referred to as the "NSS database".

The PeopleSoft system was installed in 2000 and has been modified/extended over the years to encompass:

- human resources (staff/employees, payroll, etc.)
- financials (purchases, income/expenditure, etc.)
- academic (students, courses, classes, enrolment, etc.)

At the start of October 2013, it was upgraded to Campus Solutions (CS9).

The current NSS database is separate from the UNSW Handbook, which records all of the official information related to course and program/degree requirements. To support on-line enrolment, NSS does represent some course information, such as pre-requisites, co-requisites and exclusions, as well as enrolment quotas. However, it maintains this information independently to the Handbook, which leads to potential inconsistency. Worse, NSS maintains no information at all about program/degree requirements, which means that students cannot use NSS to monitor their progress through their degree.

The MyMyUNSW database aims to implement a superset of the contents of the NSS database, including:

- people: staff, students
- infrastructure: buildings, rooms, facilities
- organisation: faculties, schools, centres
- academic: programs, streams, subjects, courses, classes

Note: there are two places in this schema where we deviate from current UNSW terminology. A **stream** is a new term that refers to what the current UNSW Handbook calls a "plan" or "specialisation". Streams also encompass the collections of courses that comprise "majors" and "minors" in many degrees. Also, in our schema, we use the terms **subject** and **course** to talk about a unit of study (subject) and a particular offering of a subject in a given semester (course). UNSW confusingly calls both of these a "course", although it also sometimes also uses the term "course offering" for the second.

The academic information includes subject pre/co-requisites and program/degree requirements, as well as all of the descriptive material from the UNSW handbook. This assemblage of information means that the MyMyUNSW database can be used as a basis for:

- on-line course and class enrolments
- managing room allocation and time-tabling
- monitoring progress through programs
- producing the UNSW Handbook

Only the first, and part of the second, of these are provided by the current PeopleSoft system.

Data Model and Schema

This section aims to give an overview of the data model. It concentrates mainly on the entities and their relationships. The details of attributes are provided by comments in the SQL schema.

Some considerations in the development of our data model:

- there are two main kinds of **people** in the system: staff and students; all people have certain basic information associated with them (e.g. name); staff have additional information related to their employment; students have additional information related to the degree that they are studying; there are also people who are neither staff nor students that UNSW wants to record (e.g. members of the University Council)
- UNSW runs a number of teaching **terms** each year (these are also called "sessions" or "semesters")
- a **subject** is a unit of study in a particular area (e.g. introductory programming, database systems, etc.); a subject is defined primarily by its syllabus
- a **course** is a particular offering of a subject in a particular teaching term; it has a course convener (also called lecturer-in-charge), perhaps an enrolment quota, and is associated with a number of classes
- a **class** is a teaching activity at a scheduled time in a scheduled place; examples of classes are lectures, tutorials and labs; a class is associated with a course
- a **degree** is an award given to a student who completes a specified program of study
- a **program** is a named program of study leading to one or more degrees
- a **stream** specifies the precise requirements for study in a specific area; it is used to implement the notions of *major* and *minor*
- in a particular program, students choose at least one stream from a range of possible streams (in a double degree, they will choose two streams, one from each set of streams for the constituent degrees); the program will specify precisely what are the allowed/required combinations of streams
- to satisfy the requirements of the program, a student must satisfy the requirements of all the streams that they enrol in from the program; in addition, there may be requirements from the program itself (e.g. general education, total units of credit completed, etc.)
- there are several different types of requirements:
  - subject requirements (core subject, electives, limitations)
  - stream requirements (e.g. must take one from the BCom majors)
  - program requirements (e.g. must be enrolled in 3648 to take SENG1010)
  - UOC requirements (e.g. overall plan needs at least 144 UC)
  - WAM requirements (e.g. must have WAM of at least 65 for Hons)
  - stage requirements (e.g. must be in stage 2 of program to take COMP2 courses)
  - stream requirements (e.g. complete one major from a set of majors)
  - miscellaneous requirements (e.g. industrial training)
- in specifying requirements, we frequently need to deal with sets of academic objects (either programs or streams or subjects); we call these (generically) **academic groupings**
- we use **subject groups** in specifying subject requirements; each subject group has a name (e.g. "level 3/4 COMP courses") and an associated set of subjects
- similarly for **stream groups** (e.g. "set of BA majors") and **program groups** (e.g. "all programs offeed by CSE")
- a **subject requirement** specifies: a subject group, a number of UOC associated with the group, whether this number is a minimum or maximum requirement; this is flexible enough to allow us to describe:
  - core requirements (group size 1, must complete 1 course from the group)
  - alternatives (several related courses, must complete 1 of them)
  - professional electives (set of courses from one area, must complete *k* of them)
  - limitations (e.g. no more than 72 UC of level 1 courses)
- in terms of the ideas above, programs and streams are defined as collections of requirements; a particular student must satisfy all of the requirements before they are regarded as having completed the program or stream
- how to determine whether a student has satisfied requirements depends on the type of requirement:
  - for UOC, use the course enrolment information
  - for course requirements, use the course enrolment information
  - for miscellaneous, must explicitly record that the student has met them (because there's no other data that will allow us to work it out)
- at any given time, each student is **enrolled** in one program, one or more streams (associated with the program), and generally several courses and classes within those courses; we need to record

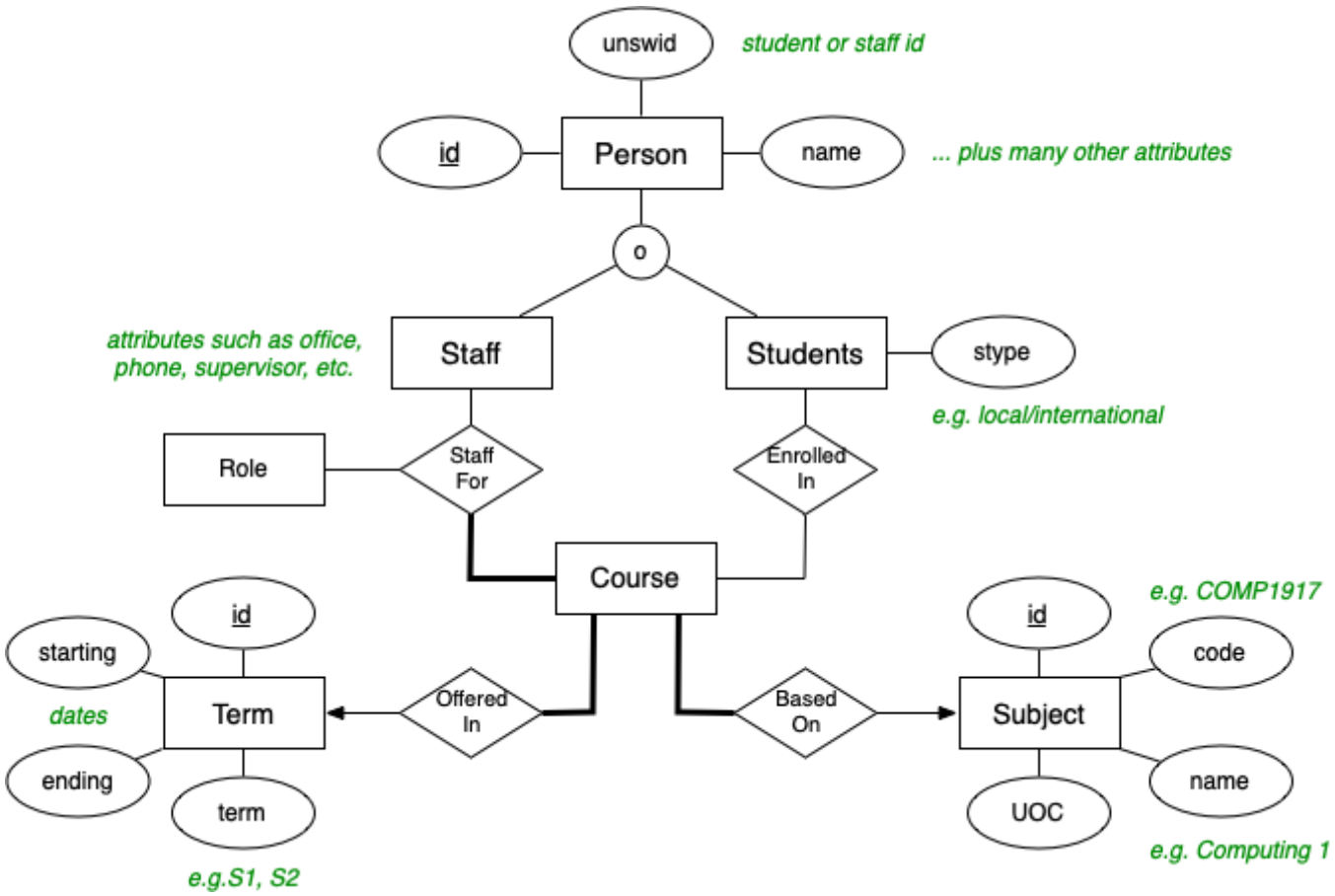
all four kinds of enrolment
- for enrolment in a program, it is useful to know when the student's enrolment commenced, when it ended (if it has ended), and their current status (e.g. active, on leave, etc.)
- over their lifetime, students may enrol in several programs, each with associated streams and courses
- a **schedule** describes when in a stream particular courses should be taken; in our terms, it will relate subject groups to streams and associate specific (year,semester) combinations with them
- sometimes, we wish to allow a student to **vary** from the standard requirements of their degree plans; there are three types of substitutions:
  - substitution: replace one course by another within a program
  - advanced standing: get credit for a course from elsewhere (or from a partly-completed UNSW degree) to use in place of some course in a program
  - exemption: get recognition for having studied a course elsewhere so that this can be used as a pre-requisite for further study at UNSW

Treat the ER data model description below as an overview, and consult the SQL Schema for full details. The ER diagrams below make a number of simplifications to the complete SQL schema: many attributes are omitted, some names are changed (e.g. entity names are singular here, but plural in the SQL schema). The SQL Schema has a detailed description of its naming conventions, which we won't repeat here. There is also a summary version of the schema available, which might be useful as a quick reference once you're familiar with the details of the schema.

To make the presentation clearer, the data model is presented in a number of sections.

People/Courses/Terms

Entities and relationships related to students/staff/courses/terms ...



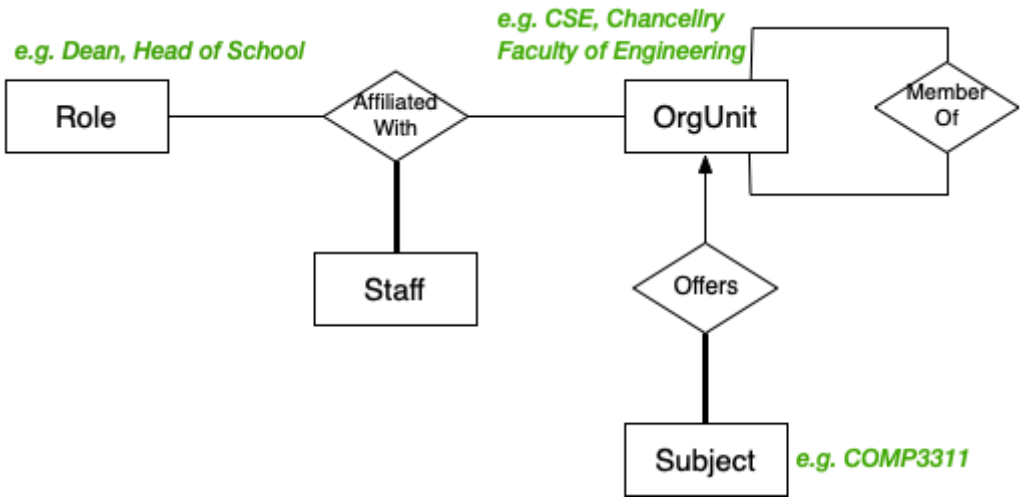
Comments:

- the Person entity would clearly have much additional data associated with it in a real implementation (contact details, etc.)
- note that the use of `Subjects.id` rather than `Subjects.code` as a primary key actually allows us to implement multiple versions of a given subject; we also need to know the period over which the particular version is relevant, and this is recorded in the subject record

- in reality, a course offering has a lot of other information associated with it (e.g. enrolment quotas, assessment schemes, classes); some of these appear in the database, some don't
- the *n:m* relationship allows multiple staff to be associated with the teaching of a given course offering; one of these staff is required to be a course convener (this could have been implemented via an extra field in the `Courses` table to force the above constraint, but since it's also a Role, we decided to implement it as such and do the constraint via a trigger)

Orgunits and Roles

Entities and relationships linking people and the unit they work in ...

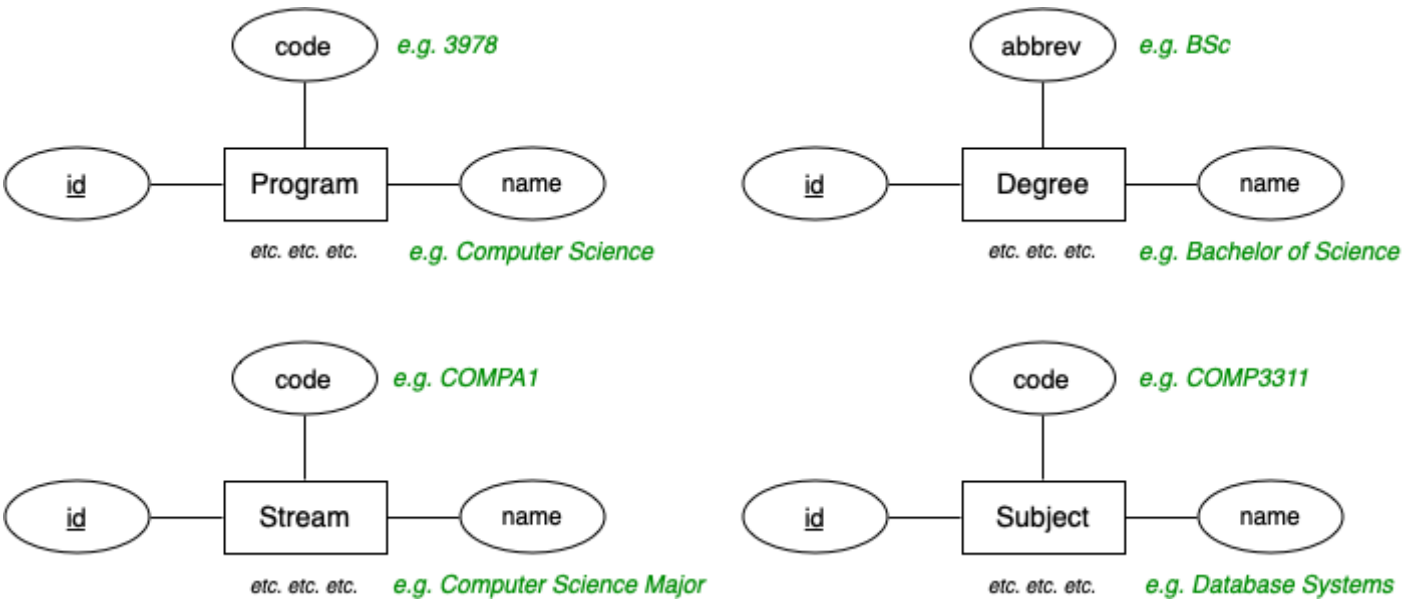


Comments:

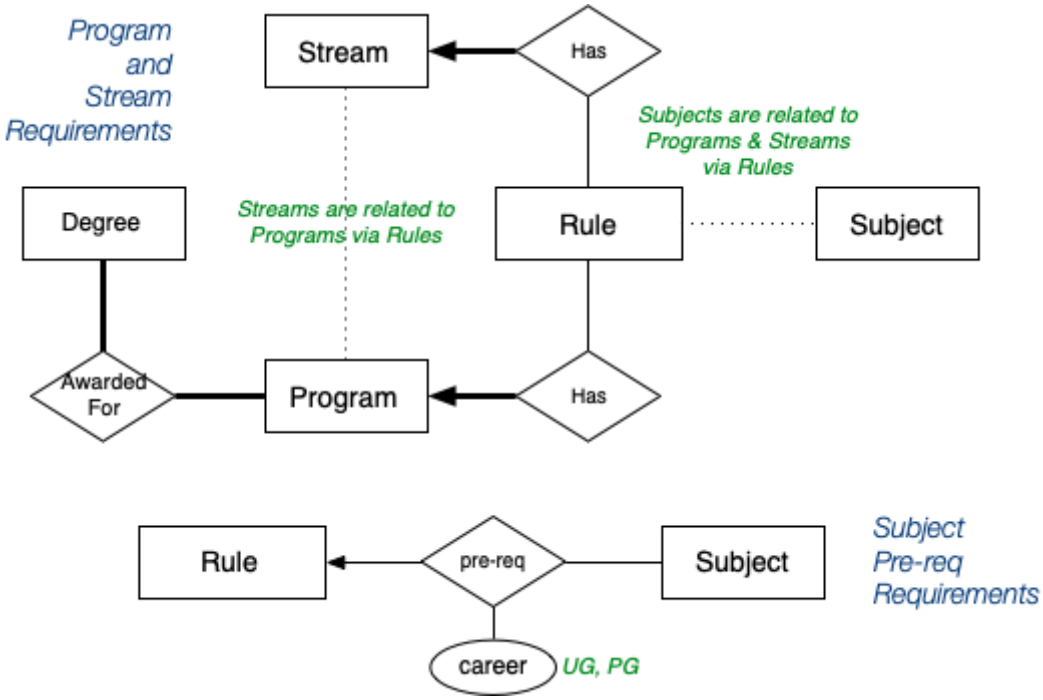
- OrgUnits = organisational units like schools, faculties, divisions
- each staff member is associated with one or more orgunits
- they may have a specific role in each orgunit
- most staff are associated with just one orgunit
- staff have a primary role, which is the basis for their employment

Programs/Streams/Degrees

Entities related to programs/streams/degrees ...



... and relationships between them ...

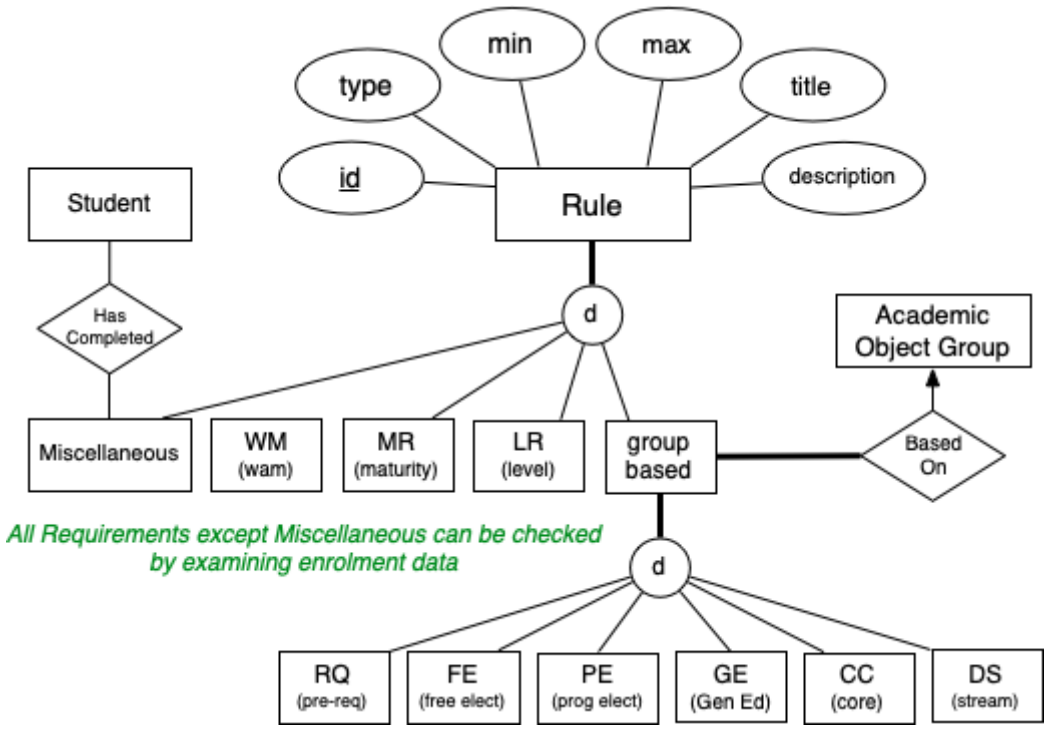


Comments:

- here, we show minimal attributes for the Program, Degree and Stream entities; see the schema for full details
- a program leads to one or more degrees (e.g. BSc, BE/BCom)
- a degree occurs in at least one program (e.g. straight BE, BE/BCom, BE/BSc)
- streams may be used in several programs (e.g. BE component of combined degrees)
- specific requirements may also be used in several plans

Rules

Entities and relationships related to requirements ...



Comments:

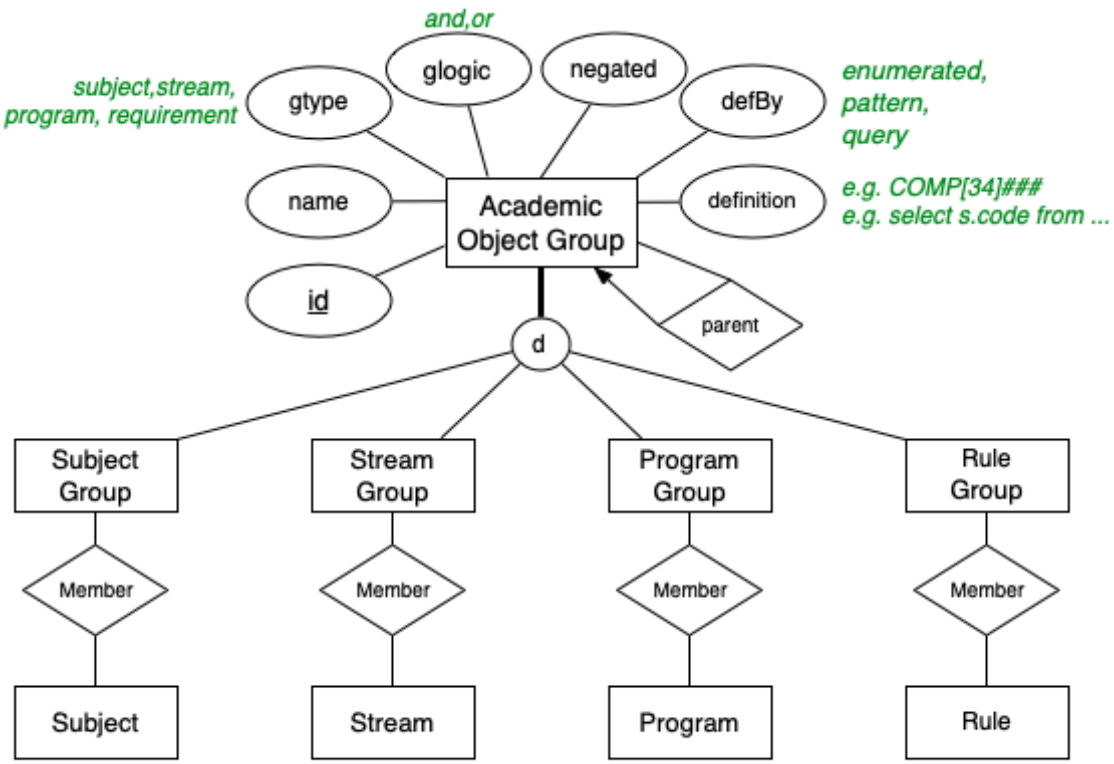
- every rule is either a
  - 'CC': core courses ... must complete min..max UOC of subjects from specified group
  - 'PE': program electives ... must complete min..max UOC of subjects from specified group
  - 'FE': free electives ... must complete min..max UOC of subjects from FREE?###
  - 'GE': general education ... must complete min..max UOC of subjects from GEN?####

  - `'RQ'`: subject pre-req ... must complete min..max UOC of subjects from group
  - `'WM'`: WAM requirement ... must achieve min WAM score
  - `'LR'`: limit rule ... must complete at least/most min/max UOC from a subject group
  - `'MR'`: maturity rule ... must complete at least min UOC before enrolling in any member of a subject group
  - `'DS'`: done stream ... must complete between min..max streams from group
  - `'RC'`: recommended ... subject group, useful for suggestions in advice
  - `'IR'`: information rule ... for information only, not checked
- since `'RC'` and `'IR'` don't have any function, we omitted them from the ER diagram
- many of the rule types refer to a set of academic objects (e.g. set of subjects in an elective group); such sets are represented by the `AcadObjectGroups` table
- a rule can be negated via a flag in the rule record
- rules may specify minimum and maximum values (see examples above)
- using min, max and academic object sets allows you express requirements like
  - must complete between 24 and 36 UOC from COMP3* courses (subject group)
  - must complete one major from the BA majors (stream group)
  - must be enrolled in a CSE degree (program group)
- if a rule is really a logical combination of other requirements, it can be expressed as a compound requirement; a compound requirement combines a set of requirements via logical AND or logical OR; compound requirements can be nested
- note that the sets of requirements for Programs, Streams amd Subject pre-requisites are implicitly conjunctive (i.e. you need to satisfy all of them before you have completed the Program or Stream or met the pre-requisite requirement for the Subject)
- miscellaneous requirements pick up all of the other non-course-related requirements that exist in various degrees (e.g. industrial training for Engineers); since there are no enrolments records or other kinds of records to check that these were completed, we need an explicit link between the student and the requirement to indicate this

Note: there are no miscellaneous or compound rules in the `mymy` database.

Academic Object Groups

Groups of academic objects that are central to the definition of requirements ...



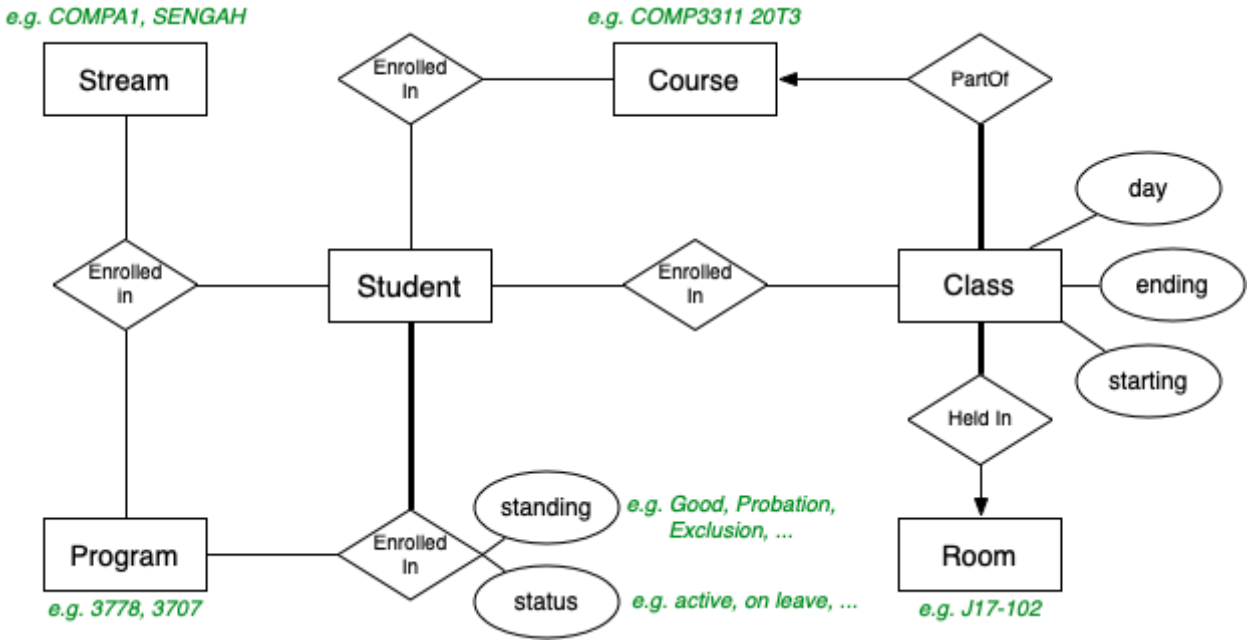
A group will have explicit members only if the parent Academic Object Group is defined by enumeration

- an academic object group (*AcObjGroup*) specifies a set of academic objects of one type (e.g. a set of streams, set of programs, etc.)

- each *AcObjGroup* has a name which describes its purpose and is used for looking up groups when programs and streams are being defined
- the members of an *AcObjGroup* may be defined in three ways:
  - by enumeration (explicitly associating each member to the group)
  - by giving a regexp pattern to identify the members (e.g. `COMP3.*`)
  - by giving an SQL query to lookup the members and return a set of IDs
- in the case of a set of requirements, you can also specify the logic of how the members are defined (conjunction or disjunction)

Enrolment

Relationships for various kinds of enrolment ...

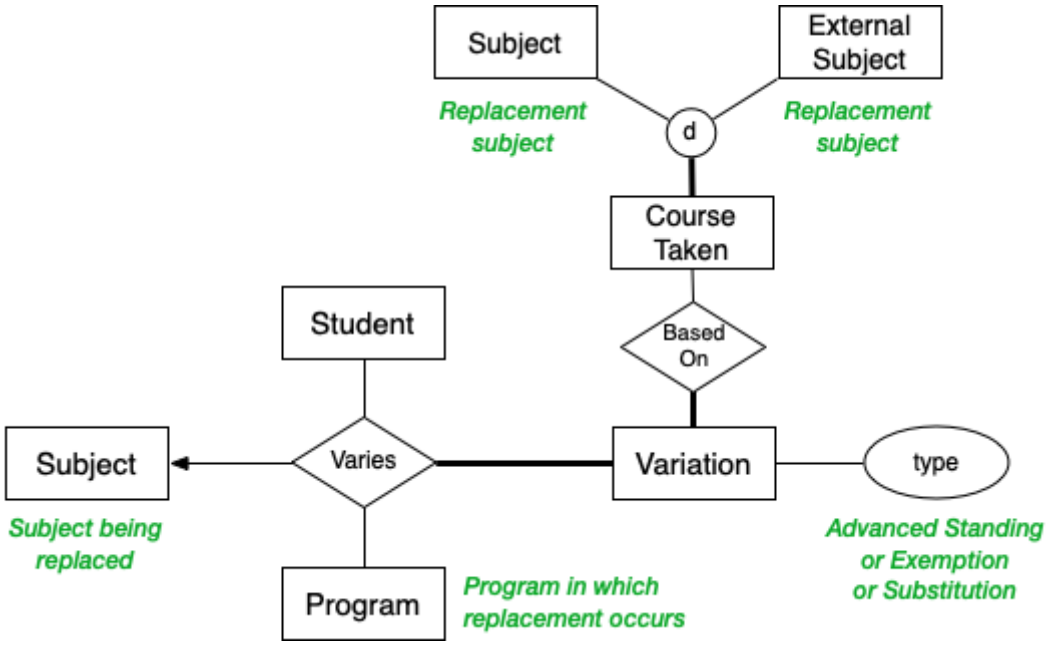


Comments:

- students initially start at UNSW by enrolling in a program
- once enrolled, they choose a specific stream (or streams) to study within that program
- this requires them to enrol in courses, and they generally enrol in one or more classes in the course (note that NSS uses enrolment in the lecture class as the way of indicating that a student is enrolled in a course; our schema does not do this)
- all these notions of "enrolment" have different kinds of information associated with them
- we have used total participation for Students-EnrollIn-Program to indicate that they must be enrolled in at least one program; it's an n:m relationship because, over time, they may enrol in other programs (e.g. complete their undergraduate degree and then later do a coursework masters degree)
- note that the mark alone is not sufficient to determine whether a student has successfully completed a course; they need a passing grade to ensure this (the set of grades indicating a pass includes SY, PC, PS, CR, DN, HD ... there is also a PT grade which is no longer used but is included to allow us to deal with old data)

Variations

Variations of meeting requirements within programs ...



Comments:

- the purpose of a variation is to indicate that a student has effectively completed one course at UNSW, without necessarily having enrolled in and passed that course
- there are three kinds of variation: substitution, advanced standing, exemption
- a substitution might be used to replace a core course in a plan by some other course if the core course is not available
- advanced standing gives credit towards a degree, based on study towards a different degree either at UNSW or elsewhere (typically, advanced standing is only granted when the degree containing the course was not completed)
- including courses from other institutions means that we need a representation for them; we have used a very simple representation
- finally, exemptions allow us to record that a student has some specific background knowledge (to use as a pre-requisite for some other course); an exemption does not confer credit towards a degree, but may be used to satisfy pre-req requirements

Note: there is no variation data in the `mymy` database.

SQL Schema

We have developed an SQL schema based on the above data model. With the above background, you are now in a good position to examine this schema, keeping in mind that the schema does not follow precisely what has been specified above in all cases.

Database Contents

The student/course/enrolment part of the database has been populated using real data which was transformed to make it anonymous (i.e. names changed to protect the innocent). The subject/program part of the database was populated by scraping the data from the UNSW Online Handbook. The class data was populated using data from the UNSW timetable site. The requirements part will be populated by hand from data in the MAPPS database. There will also be some subsequent manual tweaking of the data to ensure that there are interesting tuples to play with.

Note that the task of populating such a large database is extremely time-consuming and so many parts of the database are unpopulated or populated only very "thinly". Some parts have data that is close to reality, while others have data that is a pale shadow of reality. In order to get a feeling for what is actually present, you must spend some time exploring the database when it is eventually populated.