Assignment 2
The MyMyUNSW Database (SQL)

Last updated: **Thursday 28th October 12:46am**
Most recent changes are shown in red ... older changes are shown in brown.

Introduction

This document contains an SQL schema for the MyMyUNSW database. It implements almost all of the ER model from the Data Model page.

Schema

Downloadable Version schema.sql

```sql
--
-- COMP3311 21T3 Assignment 2 ... MyMyUNSW Schema
--

-- Useful data types

CREATE DOMAIN ShortName AS VARCHAR(16);
CREATE DOMAIN MediumName AS VARCHAR(64);
CREATE DOMAIN LongName AS VARCHAR(128);
CREATE DOMAIN ShortString AS VARCHAR(16);

CREATE DOMAIN UrlString AS VARCHAR(128) CHECK (VALUE ~ '^https?//');

CREATE TYPE AcadObjGroupDefType AS enum ('enumerated','pattern','query');

CREATE TYPE AcadObjGroupLogicType AS enum ('and','or');

CREATE TYPE AcadObjGroupType AS enum ('subject','stream','program');

CREATE TYPE CareerType AS enum ('UG','PG','HY','RS','NA');

CREATE DOMAIN CourseYearType AS INTEGER CHECK (VALUE > 1945);

CREATE TYPE GradeType AS enum
        ('A', 'A+', 'A-', 'AF', 'AS', 'AW', 'B', 'B+', 'B-', 'C', 'C+',
         'C-', 'CR', 'D', 'D+', 'D-', 'DN', 'E', 'E+', 'E-', 'EC',
         'EM', 'F', 'FL', 'HD', 'NA', 'NC', 'NF', 'PE', 'PS', 'PW',
         'RC', 'RD', 'RS', 'SY', 'UF', 'WD', 'WJ', 'XE');

CREATE TYPE TermType AS enum ('X1','S1','S2','T0','T1','T2','T3');

CREATE TYPE RuleType AS enum ('CC','PE','FE','GE','RQ','DS','MR','LR','WM')

CREATE TYPE VariationType AS enum ('advstanding','substitution','exemption'

-- Possibly useful for implementing a transcript(zid) function
CREATE TYPE TranscriptRecord AS (
CREATE TYPE TranscriptRecord AS (
        code CHAR(8),
        term CHAR(4),
        name text,
        mark INTEGER,
        grade CHAR(2),
        uoc INTEGER
);

-- Tuples from dbpop()
CREATE TYPE PopRecord AS (
        tab_name text,
        n_records INTEGER
);


-- Show database stats
CREATE FUNCTION dbpop() RETURNS SETOF public.poprecord
    LANGUAGE plpgsql
    AS $$
DECLARE
        r record;
        nr INTEGER;
        res PopRecord;
BEGIN
        FOR r IN SELECT tablename
                 FROM pg_tables
                 WHERE schemaname = 'public'
                 ORDER BY tablename
        loop
                EXECUTE 'select count(*) from '||quote_ident(r.tablename) I
                res.tab_name := r.tablename; res.n_records := nr;
                RETURN NEXT res;
        END loop;
        RETURN;
END;
$$;


-- Tables that are effectively enumerated types plus more

CREATE TABLE academic_standing (
    id INTEGER PRIMARY KEY,
    standing ShortName NOT NULL,
    notes text
);

CREATE TABLE class_types (
    id INTEGER PRIMARY KEY,
    unswid ShortString NOT NULL, -- e.g. LEC, TLB, ...
    name MediumName NOT NULL,
    description text
);

CREATE TABLE room_types (
```

```sql
CREATE TABLE Room_types (
    id INTEGER PRIMARY KEY,
    description text NOT NULL
);

CREATE TABLE Countries (
    id INTEGER PRIMARY KEY,
    code CHAR(3) NOT NULL,
    name LongName NOT NULL
);

CREATE TABLE degree_types (
    id INTEGER PRIMARY KEY,
    unswid ShortName NOT NULL,
    name text NOT NULL,
    prefix text,
    career CareerType,
    aqf_level INTEGER CHECK (aqf_level > 3)
);

CREATE TABLE stream_types (
    id INTEGER PRIMARY KEY,
    career CareerType NOT NULL,
    code CHAR(1) NOT NULL,
    description ShortString
);

CREATE TABLE orgunit_types (
    id INTEGER PRIMARY KEY,
    name ShortName NOT NULL
);

CREATE TABLE Facilities (
    id INTEGER PRIMARY KEY,
    description text NOT NULL
);

CREATE TABLE staff_roles (

    id INTEGER PRIMARY KEY,
    name text NOT NULL
);

-- Data tables

CREATE TABLE Books (
    id INTEGER PRIMARY KEY,
    isbn VARCHAR(20),
    title text NOT NULL,
    authors text NOT NULL,
    publisher text NOT NULL,
    edition INTEGER,
    pubyear INTEGER NOT NULL,
    CONSTRAINT books_pubyear_check CHECK ((pubyear > 1900))
);

```

```sql
CREATE TABLE Buildings (
    id INTEGER PRIMARY KEY,
    unswid ShortString NOT NULL,
    name LongName NOT NULL,
    gridref CHAR(4)
);

CREATE TABLE Rooms (
    id INTEGER PRIMARY KEY,
    unswid ShortString NOT NULL,
    rtype INTEGER NOT NULL REFERENCES room_types(id),
    name ShortName NOT NULL,
    fullname LongName,
    building INTEGER REFERENCES Buildings(id),
    capacity INTEGER,
    CONSTRAINT rooms_capacity_check CHECK ((capacity >= 0))
);

CREATE TABLE People (
    id INTEGER PRIMARY KEY, --zID
    family LongName,
    given LongName NOT NULL,
    fullname LongName NOT NULL,
    birthday DATE,
    origin INTEGER REFERENCES Countries(id)
);

-- simply indicates that a person is also a student
CREATE TABLE Students (
    id INTEGER REFERENCES People(id),
        PRIMARY KEY (id)
);

-- information about staff
CREATE TABLE Staff (
    id INTEGER REFERENCES People(id),
    office INTEGER REFERENCES Rooms(id),
    phone text,
    employed DATE NOT NULL,
    supervisor INTEGER REFERENCES Staff(id),
        PRIMARY KEY (id)
);

CREATE TABLE OrgUnits (
    id INTEGER PRIMARY KEY,
    utype INTEGER NOT NULL,
    name text NOT NULL,
    LongName text,
    unswid ShortString
);

CREATE TABLE Terms (
    id INTEGER PRIMARY KEY,
    YEAR CourseYearType NOT NULL,
    term CourseYearType NOT NULL,
    ttype TermType NOT NULL,
        code  CHAR(4) NOT NULL,
    name ShortName NOT NULL,
    starting DATE NOT NULL,
    ending DATE NOT NULL
);

-- Academic Objects: subjects, streams, programs

CREATE TABLE Subjects (
    id INTEGER PRIMARY KEY,
    code CHAR(8) NOT NULL,
    name MediumName NOT NULL,
    LongName LongName,
    uoc INTEGER CHECK (uoc >= 0),
    offeredby INTEGER REFERENCES OrgUnits(id),
    eftsload DOUBLE PRECISION,
    career CareerType,
    syllabus text
);

CREATE TABLE Streams (
    id INTEGER PRIMARY KEY,
    code CHAR(6) NOT NULL,
    name LongName NOT NULL,
    offeredby INTEGER REFERENCES OrgUnits(id),
    stype INTEGER REFERENCES stream_types(id),
    description text
);

CREATE TABLE Programs (
    id INTEGER PRIMARY KEY,
    code CHAR(4) NOT NULL,
    name LongName NOT NULL,
    uoc INTEGER,
    offeredby INTEGER,
    career CareerType,
    duration INTEGER,
    description text,
    CONSTRAINT programs_uoc_check CHECK ((uoc >= 0))
);

CREATE TABLE External_subjects (
    id INTEGER PRIMARY KEY,
    extsubj LongName NOT NULL,
    institution LongName NOT NULL,
    yearoffered CourseYearType,
    equivto INTEGER NOT NULL REFERENCES Subjects(id)
);

-- subject offerings
CREATE TABLE Courses (
    id INTEGER PRIMARY KEY,
    subject INTEGER REFERENCES Subjects(id),
    subject INTEGER REFERENCES Subjects(id),
    term INTEGER REFERENCES Terms(id),
    homepage UrlString
);


CREATE TABLE Classes (
    id INTEGER PRIMARY KEY,
    course INTEGER NOT NULL REFERENCES Courses(id),
    room INTEGER NOT NULL REFERENCES Rooms(id),
    ctype INTEGER NOT NULL,
    dayofwk INTEGER NOT NULL CHECK (dayofwk BETWEEN 0 AND 6),
    starttime INTEGER NOT NULL CHECK (starttime BETWEEN 8 AND 22),
    endtime INTEGER NOT NULL CHECK (endtime BETWEEN 9 AND 23),
    startdate DATE NOT NULL,
    enddate DATE NOT NULL,
    repeats INTEGER
);

CREATE TABLE Academic_object_groups (
    id INTEGER PRIMARY KEY,
    name LongName,
    TYPE AcadObjGroupType NOT NULL,
    defby AcadObjGroupDefType NOT NULL,
    definition text
);

CREATE TABLE Rules (
    id INTEGER PRIMARY KEY,
    name MediumName,
    TYPE RuleType NOT NULL,
    min_req INTEGER CHECK (min_req >= 0),
    max_req INTEGER CHECK (max_req >= 0),
    ao_group INTEGER REFERENCES Academic_object_groups(id),
    description text
);


-- Enrolment Variations
CREATE TABLE variations (
    student INTEGER REFERENCES Students(id),
    program INTEGER REFERENCES Programs(id),
    subject INTEGER REFERENCES Subjects(id),
    vtype VariationType NOT NULL,
    intequiv INTEGER,
    extequiv INTEGER,
    yearpassed CourseYearType,
    mark INTEGER CHECK (mark >= 50),
    approver INTEGER NOT NULL REFERENCES Staff(id),
    approved DATE NOT NULL,
    CONSTRAINT twocases CHECK
         ((((intequiv IS NULL) AND (extequiv IS NOT NULL)) OR
          ((intequiv IS NOT NULL) AND (extequiv IS NULL))))
);

```

```sql

-- Tables representing n:m relationships

CREATE TABLE affiliations (
    staff INTEGER REFERENCES Staff(id),
    orgunit INTEGER REFERENCES OrgUnits(id),
    ROLE INTEGER REFERENCES staff_roles(id),
    isprimary BOOLEAN,
    starting DATE NOT NULL,
    ending DATE,
        PRIMARY KEY (staff,orgunit,ROLE,starting)
);

CREATE TABLE class_enrolments (
    student INTEGER REFERENCES Students(id),
    class INTEGER REFERENCES Classes(id),
        PRIMARY KEY (student,class)
);

CREATE TABLE class_enrolment_waitlist (
    student INTEGER REFERENCES Students(id),
    class INTEGER REFERENCES Classes(id),
    applied TIMESTAMP WITHOUT TIME zone NOT NULL,
        PRIMARY KEY (student,class)
);

CREATE TABLE class_teachers (
    class INTEGER REFERENCES Classes(id),
    teacher INTEGER REFERENCES Staff(id),
        PRIMARY KEY (class,teacher)
);

CREATE TABLE course_books (
    course INTEGER REFERENCES Courses(id),
    book INTEGER REFERENCES Books(id),
    bktype VARCHAR(10) NOT NULL CHECK (bktype IN ('Text','Reference')),
        PRIMARY KEY (course,book)
);

CREATE TABLE course_enrolment_waitlist (
    student INTEGER REFERENCES Students(id),
    course INTEGER REFERENCES Courses(id),
    applied TIMESTAMP WITHOUT TIME zone NOT NULL,
        PRIMARY KEY (student,course)
);

CREATE TABLE course_enrolments (
    student INTEGER NOT NULL,
    course INTEGER NOT NULL,
    mark INTEGER CHECK (mark BETWEEN 0 AND 100),
    grade GradeType,
        PRIMARY KEY (student,course)
);

```

```
CREATE TABLE course_staff (
    course INTEGER REFERENCES Courses(id),
    staff INTEGER REFERENCES Staff(id),
    ROLE INTEGER NOT NULL
);

CREATE TABLE program_enrolments (
    id INTEGER PRIMARY KEY,
    student INTEGER NOT NULL REFERENCES Students(id),
    term INTEGER NOT NULL REFERENCES Terms(id),
    program INTEGER NOT NULL REFERENCES Programs(id),
    wam REAL,
    standing INTEGER REFERENCES academic_standing(id),
    advisor INTEGER REFERENCES Staff(id),
    notes text
);

CREATE TABLE stream_enrolments (
    partof INTEGER NOT NULL REFERENCES program_enrolments(id),
    stream INTEGER NOT NULL REFERENCES Streams(id)
);

CREATE TABLE degrees_awarded (
    student INTEGER NOT NULL,
    program INTEGER NOT NULL,
    graduated DATE
);

CREATE TABLE orgunit_groups (
    owner INTEGER REFERENCES OrgUnits(id),
    member INTEGER REFERENCES OrgUnits(id),
        PRIMARY KEY (owner,member)
);

CREATE TABLE program_degrees (
    program INTEGER REFERENCES Programs(id),
    dtype INTEGER REFERENCES degree_types(id),
    name text NOT NULL,
    abbrev text,
        PRIMARY KEY (program,dtype)
);

CREATE TABLE room_facilities (
    room INTEGER NOT NULL,
    facility INTEGER NOT NULL
);

CREATE TABLE stream_rules (
    stream INTEGER REFERENCES Streams(id),
    rule INTEGER REFERENCES Rules(id),
    PRIMARY KEY (stream,rule)
);

CREATE TABLE program_rules (
CREATE TABLE program_rules (
    program INTEGER REFERENCES Programs(id),
    rule INTEGER REFERENCES Rules(id),
        PRIMARY KEY (program,rule)
);

-- Pre-requisites
CREATE TABLE subject_prereqs (
    subject INTEGER NOT NULL REFERENCES Subjects(id),
    career CareerType NOT NULL,
    rule INTEGER NOT NULL REFERENCES Rules(id)
);


```