CS27020 DATABASE ASSIGNMENT

By Alex Smith

NOVEMBER 10, 2017

1) Elements, Data Types and Structure

Attribute Name	Data type	Example Data	Description & Constraints
Module ID	CHAR(7)	CS99910, CSM2010, IL43210	Char because it's a mix of numbers and letters, and because the length of the Module ID will always be 7. No higher, no lower.
Module Title	VARCHAR(80)	Artificial Intelligence, International Communications in Asia-Pacific: Power, Peoples and Propaganda.	VARCHAR because it can involve anything. Upper limit is 80 due to some module names being extremely long. For example, one used here is 75 characters long (From Aberystwyth University, Interpol)
Level	CHAR	1, M, 3	The level is always only a single space long. It can be either a letter or number, so a char is perfect here.
Class Size	Int	240, 10	Class size will never be larger than 999 people (I should really, really hope not) so I can limit it to 3. The data will always be numerical so an Integer is appropriate here.
Time	Time	13:00, 09:00	The Time type is rather good at storing time, so It was used here. Constraints: No lectures before 09:00 or after 17:00, apart from Wednesday -cannot be after 12:00.
Day	VARCHAR(9)	Wednesday, Monday	Days have different lengths but a maximum of 9 (Wednesday is the longest name at 9). A VARCHAR will store these days with a character limit of 9. Constraint: No lectures on weekends!
Campus	VARCHAR(30)	Geenfields, Science Park	A limit of 30 characters seems to allow for even the longest campus names and accounts for any new campus.
Building	VARCHAR(40)	Science Building, Sir James watt	Some buildings can have extremely long names (commemorative buildings, for example). 40 will account for all of this.
Room	VARCHAR(20)	A3, Main laboratory	20 seems an appropriate amount of space to store Room names. Main laboratory is 15 characters, so 20 seems to fit it well.
Style	VARCHAR(20)	Lecture Theatre, Seminar room	Similar to Room name. 20 fits 'Lecture theatre' easily, which is the longest given so far.
Seating capacity	Int	320, 70	A lecture room will never exceed 999 seats (indeed, the largest from the data is less than a third of this). An Integer will be suitable for this, and stop non-numerical input.

2) Functional Dependencies

Module ID -> Title, Level, Class Size:

- Module ID contains information on the Title and level, and you can obtain the class size also.

Building -> Campus

- Buildings are unique so you can find the campus name.

Room -> Building, Campus, Style, Seating capacity

- Room names are unique, so you can find the Building and therefore Campus. Additionally, the Style and Seating Capacity are attributes of Room, so can be found too.

Nothing is dependent on the Module Title, Level, Time, Day, Campus, Style, Seating capacity and Class size.

3) Normalisation

UNF

Module ID	Module Title	Level	Time	Day	Campus	Building	Room	Style	Seating Capacity	Class Size
CS 99910	Artificial Intelligence	1	13:00	Thursday	City Campus	Stapleton	B23	Computer Room	320	240
CSM2010	Artificial Intelligence	M	13:00	Thursday	City Campus	Stapleton	B23	Computer Room	320	10
HH22820	Roman Empire	2	9:00 14:00	Tuesday Friday	City Campus	Stapleton	B40 A3	Seminar Room Lecture Theatre	70 85	56
GG23330	Volcanology	3	10:00 11:00 17:00	Tuesday Tuesday Thursday	Greenfields Greenfields Science Park	Sir James Watt Sir James Watt Science Building	Main Lab Main Lab Lecture 1	Laboratory Laboratory Lecture Theatre	50 50 60	43
IL43210	Information Retrieval	3	14:00	Wednesday	Science Park	Physics	1.45	Laboratory	75	63
LA28320	Constitutional Law	1	16:00	Wednesday	City Campus	Stapleton	A14	Lecture theatre	200	103
CS70005	Compiler Design	2	15:00	Monday	City Campus	Stapleton	B23	Computer Room	320	95

First step was to combine the data in to a table – this was to help fill in the blanks of what data I still needed.

1NF

To get to 1st Normal Form I needed to split non-atomic values. Then, split the tables into meaningful relations. For 1NF I chose to use these three because they make the most logical sense. This meant that Time and Day had to be separated out. To do so, I had to include the primary keys of the other two tables to make it composite.

Module ID	Title	level	Class Size
CS 99910	Artificial Intelligence	1	240
CSM2010	Artificial Intelligence	M	10
HH22820	Roman Empire	2	56
GG23330	Volcanology	3	43
IL43210	Information Retrieval	3	63
LA28320	Constitutional Law	1	103
CS70005	Compiler Design	2	95

Room	Building	Campus	Style	Seating Capacity
B23	Stapleton	City Campus	Computer Room	320
B40	Stapleton	City Campus	Seminar Room	70
A3	Stapleton	City Campus	Lecture Theatre	85
Main Lab	Sir James Watts	Greenfields	Laboratory	50
Lecture 1	Science Building	Science Park	Lecture Theatre	60
1.45	Physics	Science Park	Laboratory	75
A14	Stapleton	City Campus	Lecture Theatre	200

Time	Day	Module ID (FK)	Room (FK)
13:00	Thursday	CS99910	B23
13:00	Thursday	CSM2010	B23
09:00	Tuesday	HH22820	B40
14:00	Friday	HH22820	A3
10:00	Tuesday	GG23330	Main Lab
11:00	Tuesday	GG23330	Main Lab
17:00	Thursday	GG23330	Lecture 1
14:00	Wednesday	IL43210	1.45
16:00	Wednesday	LA28320	A14
15:00	Monday	CS70005	1.45

2NF

No changes need to be made to my tables for 2nd Form to apply. For example, we need all of the attributes of the 3rd table to find anything. It would not be possible from just Time and Day, or Time and ModuleID. As for partial dependencies, there can be none because the first two tables only have a single Primary Key. In the third table, each attribute is a part of the composite primary key, so there can be no attribute that can be a primary key by itself.

Module ID	Title	level	Class Size
CS 99910	Artificial Intelligence	1	240
CSM2010	Artificial Intelligence	M	10
HH22820	Roman Empire	2	56
GG23330	Volcanology	3	43
IL43210	Information Retrieval	3	63
LA28320	Constitutional Law	1	103
CS70005	Compiler Design	2	95

Room	Building	Campus	Style	Seating Capacity
B23	Stapleton	City Campus	Computer Room	320
B40	Stapleton	City Campus	Seminar Room	70
A3	Stapleton	City Campus	Lecture Theatre	85
Main Lab	Sir James Watts	Greenfields	Laboratory	50
Lecture 1	Science Building	Science Park	Lecture Theatre	60
1.45	Physics	Science Park	Laboratory	75
A14	Stapleton	City Campus	Lecture Theatre	200

Time	Day	Module ID (FK)	Room (FK)
13:00	Thursday	CS99910	B23
13:00	Thursday	CSM2010	B23
09:00	Tuesday	HH22820	B40
14:00	Friday	HH22820	A3
10:00	Tuesday	GG23330	Main Lab
11:00	Tuesday	GG23330	Main Lab
17:00	Thursday	GG23330	Lecture 1
14:00	Wednesday	IL43210	1.45
16:00	Wednesday	LA28320	A14
15:00	Monday	CS70005	1.45

3NF

Room	Building (FK)	Style	Seating Capacity
B23	Stapleton	Computer Room	320
B40	Stapleton	Seminar Room	70
A3	Stapleton	Lecture Theatre	85
Main Lab	Sir James Watts	Laboratory	50
Lecture 1	Science Building	Lecture Theatre	60
1.45	Physics	Laboratory	75
A14	Stapleton	Lecture Theatre	200

Module ID	Title	level	Class Size
CS99910	Artificial Intelligence	1	240
CSM2010	Artificial Intelligence	M	10
HH22820	Roman Empire	2	56
GG23330	Volcanology	3	43
IL43210	Information Retrieval	3	63
LA28320	Constitutional Law	1	103
CS70005	Compiler Design	2	95

Building	Campus
Stapleton	City Campus
Sir James Watts	Greenfields
Science Building	Science Park
Physics	Science Park

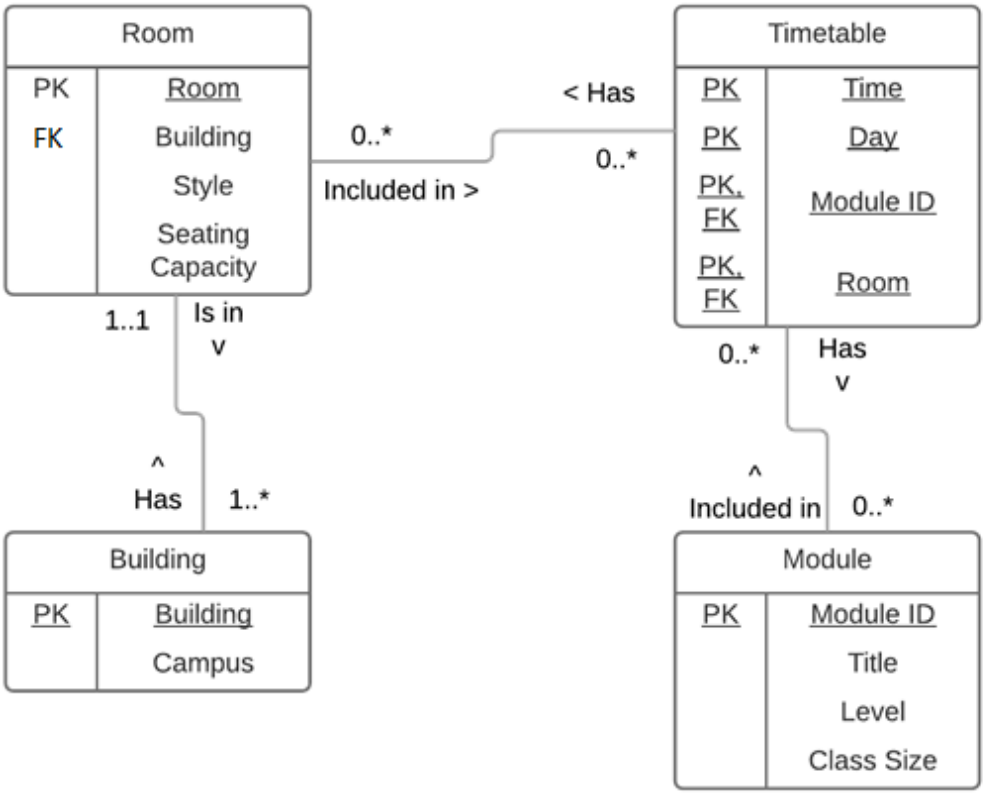
Time	Day	Module ID (FK)	Room (FK)
13:00	Thursday	CS99910	B23
13:00	Thursday	CSM2010	B23
09:00	Tuesday	HH22820	B40
14:00	Friday	HH22820	A3
10:00	Tuesday	GG23330	Main Lab
11:00	Tuesday	GG23330	Main Lab
17:00	Thursday	GG23330	Lecture 1
10:00	Wednesday	IL43210	1.45
11:00	Wednesday	LA28320	A14
15:00	Monday	CS70005	1.45

For Third Normal Form the Module ID relation does not need to be changed as the other attributes are non-prime and therefore unable to find each other.

The Room relation, however, has the building and campus attributes. From Building (a non-prime attribute), we can find campus (another non-prime attribute). This means that Campus and Building must be taken from the relation and placed in their own. To do so I had to remove Building and Campus from their original table, and place them in a new one. Then I needed to create a foreign key of Building in the Room table, to ensure it was linked.

Note: You could pull Campus from Building table and place a Campus FK in the Building table. This is so you can change/create new campuses whenever. This could be done in other cases.

4) ER diagram



A room number will be included in a timetable entry. There may be none (for blank periods/empty timetables) to many, as you may have different rooms in a day. Likewise, a timetable entry will have none to many rooms timetabled for that day/time.

The room table will have Only one building to any one room. The building table is one to many as a building can have anything from a single room to many.

A timetable will have none to many modules as it may go from being un-populated to having several modules planned in a day or week. Modules will be included in none to many timetables as no one may take a module, or several different degree schemes will have the same module (from the example data: Artificial science is for both level M and 1.

5) Creating and populating tables

Creating the tables:

```
CREATE TABLE Building_Relation (
  Building VARCHAR(40) NOT NULL,
  Campus VARCHAR(30) NOT NULL,
  PRIMARY KEY(Building)
  );

CREATE TABLE Room_Relation (
  Room VARCHAR(20) NOT NULL,
  Building VARCHAR(40) NOT NULL,
  Style VARCHAR(20) NOT NULL,
  Seating_Capacity INT NOT NULL,
  PRIMARY KEY(Room),
  FOREIGN KEY(Building) REFERENCES Building_Relation (Building)
);

CREATE TABLE ModuleID_Relation (
  ModuleID CHAR(7) NOT NULL,
  Module_Title VARCHAR(80) NOT NULL,
  Level CHAR NOT NULL,
  Class_Size INT NOT NULL,
  PRIMARY KEY(ModuleID)
);

CREATE TABLE Timetable_Relation (
  ModuleID CHAR(7) NOT NULL,
  Room VARCHAR(20) NOT NULL,
  Times TIME NOT NULL,
  Day VARCHAR(9) NOT NULL,

  CHECK (Day = 'Monday' OR Day = 'Tuesday' OR Day = 'Wednesday' OR Day = 'Thursday' OR Day =
'Friday'),

  CONSTRAINT Hour_Times CHECK ((EXTRACT(HOUR FROM Times) >=9) AND (EXTRACT(HOUR FROM
Times)<=17)),
  CONSTRAINT Wed_Times CHECK (NOT (day = 'Wednesday' AND (EXTRACT(HOUR FROM Times) >= 12))),

  PRIMARY KEY (Times, Day, ModuleID, Room),
  FOREIGN KEY (ModuleID) REFERENCES ModuleID_Relation (ModuleID),
  FOREIGN KEY (Room) REFERENCES Room_Relation (Room)
);
```

Inserting data into the tables:

```
INSERT INTO Building_Relation(Building, Campus) VALUES
  ('Stapleton' , 'City Campus'),
  ('Sir James Watts' , 'Greenfields'),
  ('Science Building' , 'Science Park'),
  ('Physics' , 'Science Park')
;

INSERT INTO Room_Relation(Room, Building, Style, Seating_Capacity) VALUES
  ('B23' , 'Stapleton' , 'Computer room' , '320'),
  ('B40' , 'Stapleton' , 'Seminar Room' , '70'),
  ('A3' , 'Stapleton' , 'Lecture Theatre' , '85'),
  ('Main Lab' , 'Sir James Watts' , 'Lab' , '50'),
  ('Lecture 1' , 'Science Building' , 'Lecture Theatre' , '60'),
  ('1.45' , 'Physics' , 'Laboratory' , '75'),
  ('A14' , 'Stapleton' , 'Lecture Theatre' , '200')
;

INSERT INTO ModuleID_Relation(ModuleID, Module_Title, Level, Class_Size) VALUES
  ('CS99910' , 'Artificial Intelligence' , '1' , '240'),
  ('CSM2010' , 'Artificial Intelligence' , 'M' , '10'),
  ('HH22820' , 'Roman Empire' , '2' , '56'),
  ('GG23330' , 'Volcanology' , '3' , '43'),
  ('IL43210' , 'Information Retrieval' , '3' , '63'),
  ('LA28320' , 'Constitutional Law' , '1' , '103'),
  ('CS70005' , 'Compiler Design' , '2' , '95')
;
```

```
INSERT INTO Timetable_Relation(Times, Day, ModuleID, Room) VALUES
  ('13:00:00' , 'Thursday' , 'CS99910' , 'B23'),
  ('13:00:00', 'Thursday',  'CSM2010', 'B23'),
  ('09:00:00', 'Tuesday', 'HH22820', 'B40'),
  ('14:00:00', 'Friday' , 'HH22820' ,  'A3'),
  ('10:00:00', 'Tuesday' ,  'GG23330'  , 'Main Lab'),
  ('11:00:00', 'Tuesday' ,  'GG23330' ,  'Main Lab'),
  ('17:00:00', 'Thursday' , 'GG23330' ,  'Lecture 1'),
  ('10:00:00', 'Wednesday' ,  'IL43210' ,  '1.45'),
  ('11:00:00', 'Wednesday' ,  'LA28320' ,  'A14'),
  ('15:00:00', 'Monday' , 'CS70005' ,  '1.45')
;
```

6) Query's

Query one:

```
SELECT
  Timetable_Relation.times,
  Timetable_Relation.day,
  Timetable_Relation.Room,
  Timetable_Relation.moduleid,
  Room_Relation.building,
  Building_Relation.campus
-- this includes last two as data is for a timetable- suits question/query
FROM Timetable_Relation

INNER JOIN Room_Relation ON Timetable_Relation.Room = Room_Relation.Room
INNER JOIN Building_Relation ON Room_Relation.Building = Building_Relation.Building

WHERE moduleID = 'GG23330'
```

Query two:

```
SELECT Timetable_Relation.moduleID
FROM Timetable_Relation

INNER JOIN Room_Relation ON Timetable_Relation.Room = Room_Relation.Room
INNER JOIN Building_Relation ON Room_Relation.Building = Building_Relation.Building

GROUP BY timetable_relation.moduleID

HAVING COUNT (DISTINCT Campus) > 1
```

Query three:

```
--list all classes that have less than 75% of pupils
SELECT
  Timetable_Relation.times,
  Timetable_Relation.day,
  Timetable_Relation.Room,
  sum(class_size),
  seating_capacity


FROM timetable_relation

INNER JOIN Room_Relation on timetable_relation.room = room_relation.room
INNER JOIN moduleid_relation ON timetable_relation.moduleid = moduleid_relation.moduleid
INNER JOIN Building_Relation ON Room_Relation.Building = Building_Relation.Building

GROUP BY moduleid_relation.class_size, room_relation.seating_capacity, Timetable_Relation.times,
Timetable_Relation.day, Timetable_Relation.Room
HAVING CAST(sum(class_size) AS FLOAT)/CAST(seating_capacity AS FLOAT)<0.75
```

7) Self evaluation

Section one: Here I believe I comprehensibly accounted for all data types/storage, limits and constraints on the data.

Section two: The functional dependencies in my tables were accurately named and adequately described. They allowed me to create my tables to a higher standard.

Section three: My normalisation took the longest due to the varying descriptions of what each NF actually is. Much revision took place here and I believe I got them correct.

Section four: The ER diagram did not take too long and I believe I achieved full marks here due to the clear layout.

Section five and six: My implementation was not the most efficient and rather basic but I am sure that the results will show that my tables set up well and my query's ask for the desired data.