SAMPLE PROBLEM: NORMALIZATION

COMP 1140: Database and SQL | Brendan Shea, PhD(Brendan.Shea@rctc.edu)

In this lesson, we'll go over a sample database normalization problem. Suppose that we are presented with a spreadsheet that contains data of the following form:

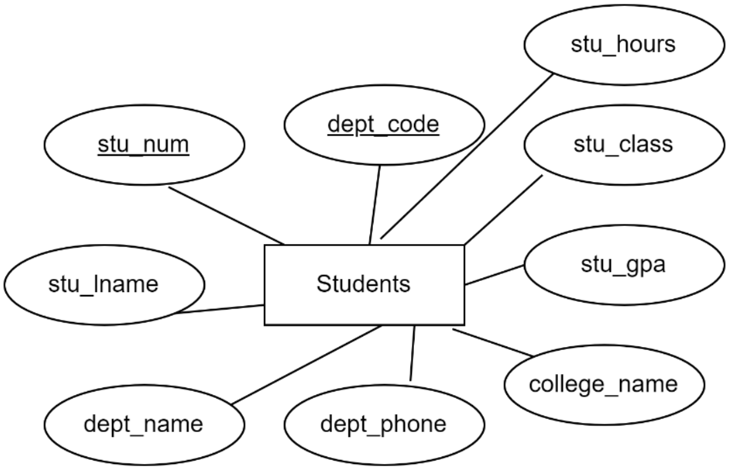
Table: Students and their Majors

stu_num	stu_lname	dept_code	dept_name	dept_phone	college_name	stu_gpa	stu_hours	stu_class
013728	Joyce	CS	Comp Science	4356	Engineering	3.46	75	Junior
248339	Beckett	CS	Comp Science	4356	Engineering	3.94	45	Sophomore
255620	Behan	ART	Art	4378	Liberal Arts	3.45	117	Senior
180483	Wilde	ART	Art	4378	Liberal Arts	2.42	113	Senior
426821	Yeats	PHIL, CS	Philosophy, Computer Science	3520, 4356	Liberal Arts, Engineering	2.72	87	Junior
079852	Parnell	MATH	Mathematics	3420	Liberal Arts	2.9	89	Junior

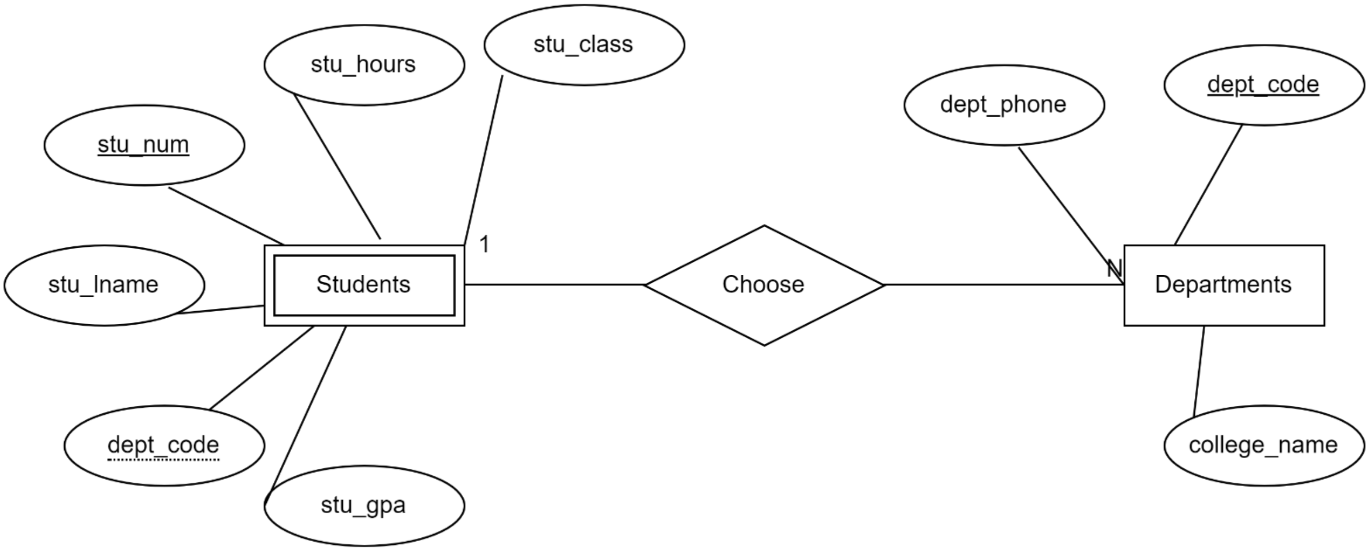
1. Write down the **relational schema,**
   a. Students(stu_num, stu_lname, dept_code, dept_name, dept_phone, college_name, stu_gpa, stu_hours, stu_class)
2. Is the table in 1NF? If it is not, identify any **repeating groups** that exist, and show how you could resolve them.
   a. No. One row (Yeats) contains a repeating group, presumably because this student is a double major. The quickest way to resolve this would to simply separate these out into two rows:

stu_num	stu_lname	dept_code	dept_name	dept_phone	college_name	stu_gpa	stu_hours	stu_class
013728	Joyce	CS	Comp Science	4356	Engineering	3.46	75	Junior
248339	Beckett	CS	Comp Science	4356	Engineering	3.94	45	Sophomore
255620	Behan	ART	Art	4378	Liberal Arts	3.45	117	Senior
180483	Wilde	ART	Art	4378	Liberal Arts	2.42	113	Senior
426821	Yeats	PHIL	Philosophy	3520	Liberal Arts	2.72	87	Junior
426821	Yeats	CS	Comp Science	4356	Engineering	2.72	87	Junior
079852	Parnell	MATH	Mathematics	3420	Liberal Arts	2.9	89	Junior

3. Identify any **candidate keys.** Explain your reasoning.
   a. While stu_num *looks* like a candidate key, it can't be one (since there are two rows with the same student key). However, the combination of (**stu_num, dept_code**) is a candidate key.
   b. This key is a "superkey" fully determines all of the other attributes. That is, if we knew the key, we could easily "retrieve" any row from the table
   c. It is also a "minimal superkey"—we don't need any other attributes.
4. Draw the ER diagram so far.

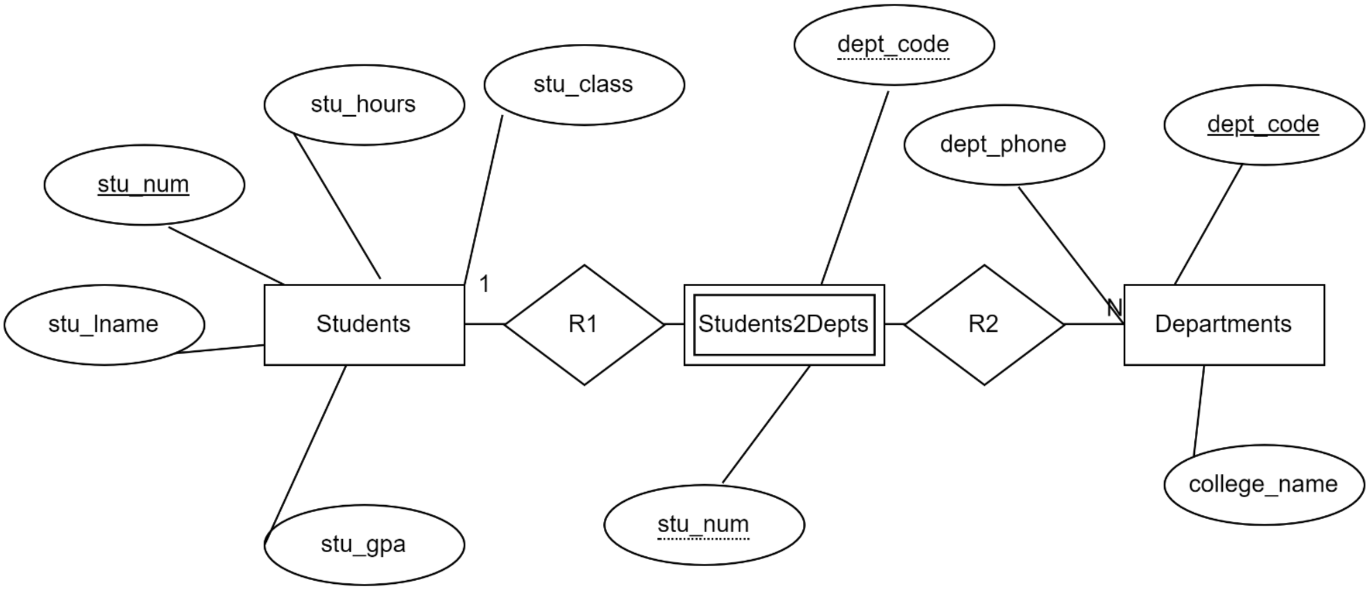


5. Is this database in 2NF? If yes, explain why. If no, identify any **partial dependencies,** and explain how to resolve them. Give the new relational schema.
    a. It is NOT in 2NF, since we have a composite key and some of our attributes are dependent on only part of that key.
    b. Partial dependency: stu_num -> (stu_lname & student_gpa & stu_hours).
        i. In other words, knowing a student's id number will let you determine their name, gpa, and credits completed.
    c. Partial dependency : dept_code -> (dept_nam & dept_phone & college_name)
        i. Knowing dept_code allows you to determine the department name and phone number AND the college
    d. Solution: Separate dept info from student info, with students having a one-to-many relationship to departments:
        i. Students(stu_num, stu_lname, stu_major, college_name, stu_gpa, stu_hours, stu_class, dept_code (FK))
        ii. Departments(dept_name, dept_phone, college_name)
6. Draw the ER diagram so far.



The Problem: This design is far from perfect, as it leaves Students with a composite key that drawn in part from Departments. This makes Students weak, which seems odd/wrong! Among other things, this would make it tough to enter students who haven't chosen majors yet! For this reason, in real life, we'd probably want to model this as THREE tables: Students (with just stu_num as key), Departments, and the join table Students2Department.

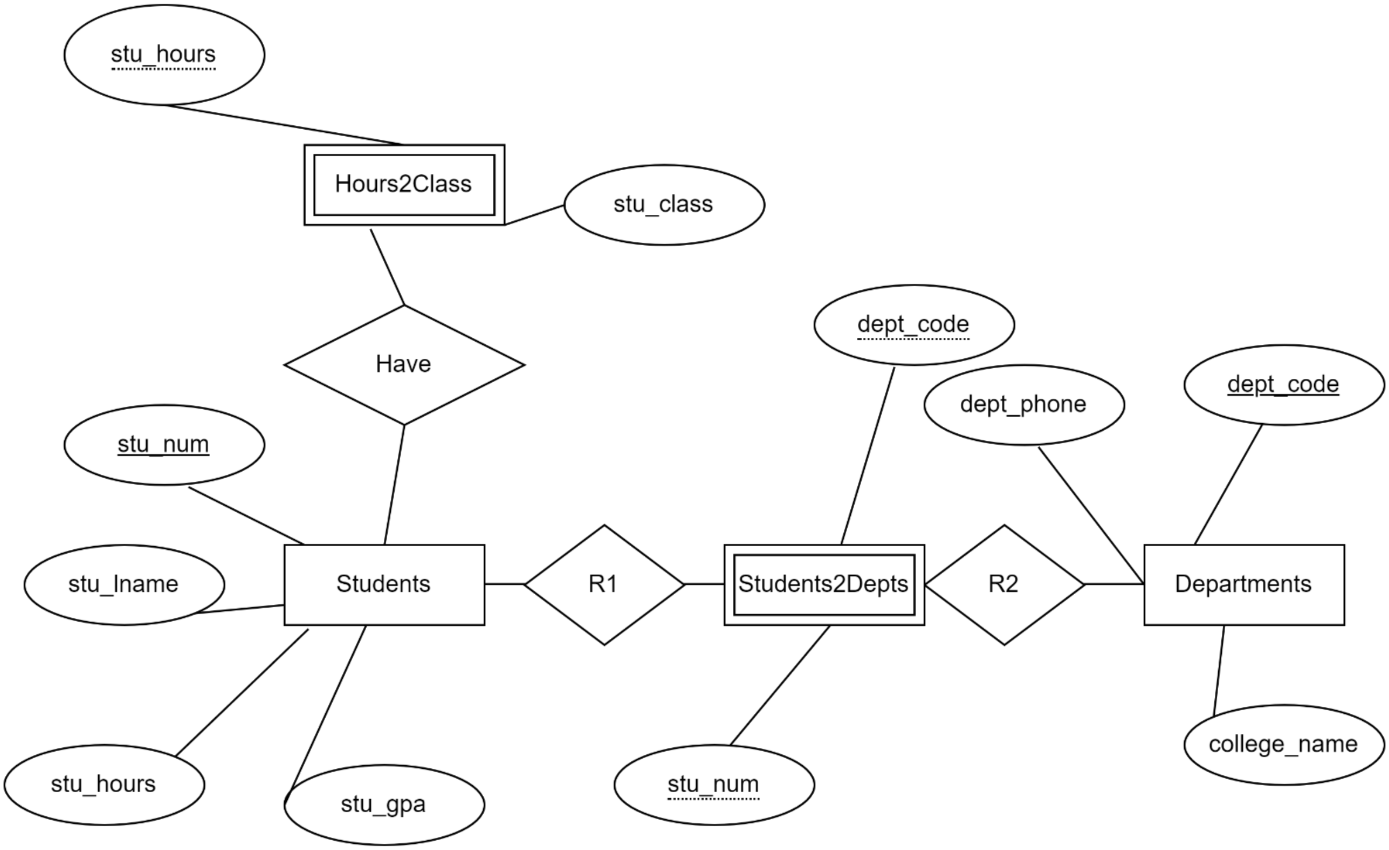
This would look like the following:



7. Is this database in 3NF? If yes, explain why. If no, identify any transitive **dependencies,** and explain how to resolve them. Give the new relational schema.
   a. It is NOT in 3NF, since there is a transitive dependency.
   b. In particular, in the Students table, stu_classification (as freshmen, junior, senior) is dependent on the primary key only THROUGH the non-key attribute stu_hours (a measure of how many credits a student has completed) allows you to determine.
   c. Theoretically, this could lead to inconsistencies regarding how student classification (e.g., students with the same number of credit hours could be classified differently).
   d. Solution: Create a weak "range table" **Hours2Classification** that translates student hours to the appropriate classification. Remove classification from Students.
   e. In real-life, you may not do this! There's very little cost to simply storing the classification directly along with the student record (and thus, avoiding doing the lookup every time). Moreover, it is relatively *unlikely* that we'll make changes to the look up table (of the sort that might lead to redundancies or anomalies). That being said, it might be better safe than sorry (if, for example, we might eventually want to further subdivide classification for registration purposes, etc.).

min_credits	max_credits	Classification
0	32	Freshmen
33	64	Sophomore
65	96	Junior
97	MAX_INT	Senior

8. Draw the ER diagram in 3NF.


stu_hours
Hours2Class
stu_class
Have
dept_code
dept_phone
dept_code
stu_num
stu_lname
Students
R1
Students2Depts
R2
Departments
stu_hours
stu_gpa
stu_num
college_name