# ECE 656 Winter 2019: Assignment 2

**(1)** Consider the following schema. It is a small database for a university course scheduling system, providing details about courses, offerings, instructors, classrooms and departments. Neither primary keys nor foreign keys have been specified in this description, and that is deliberate. The table name (relation) is in **bold**.

**Instructor** (instID, instName, deptID, sessional)
**Course** (courseID, courseName, deptID, prereqID)
**Offering** (courseID, section, termCode, roomID, instID, enrollment)
**Classroom** (roomID, building, room, capacity)
**Department** (deptID, deptName, faculty)
Explanation:

- **Instructor** defines a unique instructor ID, his/her name, department and sessional status.
- **Course** defines a unique course ID, the course name, the department offering that offers the course, and any prerequisites.
- **Offering** defines an actual offering of a course; the offering comprises the courseID being offered, the section number (integers starting at 1), the term code (the standard UW 4-digit term code: the first three digits define the year (add 1900 to get the year), and the fourth digit is the month in which the course starts (1 (Jan), 5 (May), or 9 (Sept) for Winter, Spring, and Fall offerings, respectively), the room where the section meets, the instructor, and the number of students enrolled.
- **Classroom** defines a unique room ID, together with the building, room number and room capacity.
- **Department** identifies a unique department ID and its name.

A subset of the data is as shown below:

**Instructor**

| instID int | instName char(10) | deptID char(4) | sessional bool |
|---|---|---|---|
| 1 | Nelson | ECE | false |
| 3 | Jimbo | ECE | false |
| 4 | Moe | CS | true |
| 5 | Lenny | CS | false |

**Course**

| courseID char(8) | courseName varchar(50) | deptID char(4) | prereqID char(8) |
|---|---|---|---|
| ECE356 | Database Systems | ECE | ECE250 |
| ECE358 | Computer Networks | ECE | ECE222 |
| ECE390 | Engineering Design | ECE | ECE290 |
| MATH117 | Calculus 1 | MATH | null |

**Offering**

| courseID char(8) | section int | termCode decimal(4) | roomID char(8) | instID int | enrollment int |
|---|---|---|---|---|---|
| ECE356 | 1 | 1191 | E74417 | 1 | 64 |
| ECE356 | 2 | 1191 | E74417 | 3 | 123 |
| ECE358 | 2 | 1191 | E74417 | 1 | 123 |
| ECE390 | 1 | 1191 | E74053 | 1 | 102 |
| MATH117 | 1 | 1189 | RCH111 | 5 | 134 |

**Classroom**

| roomID char(8) | Building char(4) | Room dec(4) | Capacity int |
|---|---|---|---|
| E74417 | E7 | 4417 | 138 |
| E74053 | E7 | 4053 | 144 |
| RCH111 | RCH | 111 | 91 |
| RCH101 | RCH | 101 | 250 |

**Department**

| deptID char(8) | deptName varchar(50) | faculty varchar(50) |
|---|---|---|
| ECE | Electrical and Computer Engineering | Engineering |
| CS | Computer Science | Math |
| MATH | Math | Math |
| C&O | Combinatorics and Optimization | Math |

**(a)** What are plausible Primary Keys on each of the five relations?

**(b)** What are plausible Foreign Keys for the five relations?

**(c)** What additional constraints, if any, should be added?

**(d)** Knowing that each department is part of a faculty (deptID $\rightarrow$ faculty), that courses can have more than one prerequisite, and desiring to be able to do queries based on term (Winter, Spring, Fall) without regard to the particular year (*e.g.*, what courses are offered in the fall term?), what modifications to the schema, if any, are needed to ensure that it is either 3NF or BCNF (your choice)? If there are any new of changed relations, identify them, including any changes or adjustments to primary keys and/or foreign keys, or any other constraints. Explain your reasoning.

**(e)** Considering queries for the following purposes:

- Which instructors are sessionals?
- Which instructors have taught courses over a particular timeframe?
- How many courses are taught by sessionals?
- How many students are taught by sessionals?
- Any of the above, grouped by faculty
- Any of the above, as fractions of total instructors and/or courses, as relevant?

Using "explain" and/or your own reasoning, identify what indexes would be potentially useful to help in these queries.

**(f)** Considering the specific query:

```
select count(courseID) from Course inner join Department using (deptID)
                            where prereqID is NULL and faculty='Math';
```

Assuming no indexes, what would the execution plan be and what would be the estimated execution time for that plan if the tables are on disk, in contiguous blocks, the number of rows in Course is $r_c$, the number of blocks in Course is $b_c$, the number of rows in Department is $r_d$, the number of blocks in Department is $b_d$, the time to find a random block on disk is $T_s$ and the time to transfer a block from disk is $T_t$? **(g)** For

the query above, identify any indexes over one or more attributes that might potentially improve the query performance. For each index you identify, specify the type of index (B+-tree or Hash or either), whether or not it is a primary or secondary index, if it is a secondary index identify if it is useful if it is an index extension, and justify why the query might benefit from that index.

**(2)** In assigment 1 you had to compute several queries on the Sean Lahman baseball database. There were no explicit indexes on that database, though you should have added primary and foreign keys. Using explain on the queries you created for Lab 1, determine if any additional explicit indexes would help in solving those queries.

**(3)** Likewise, you had to compute several queries on the Yelp database. Again, using explain on the queries you created for Lab 1, determine what indexes would help in solving those queries.