CSC 1035
ERD's – packet 4: more on ERD<->RDB connections

a) Packet 3 illustrated some weak entity classes and how they’re mapped to an RDB. **How can that be formulated as step 1.1 in the recipe for mapping an ERD to an RDB?**

b) Sometimes a weak entity class needs not just one, but two defining strong classes. This leads to an ERD like this one:

![Entity Relationship Diagram](image)

Note that the “Scheduled” relationship is ternary, it’s rounded (since Appointments is defined through both Patients and Doctors), and we haven’t classified cardinalities for it.

Here’s the RDB it leads to:

- Patients(patID, pName)
- Appointments(patID, drID, TimeDate, cost)
- Doctors(drID, drName)

**How can this situation be formulated as step 1.2 in the recipe for mapping an ERD to an RDB?**
c) Look at the UniversityDB.accdb example we’ve been using. Here’s how it might be “reverse-engineered” to an ERD (easy):

Notice that facID isn’t included as an attribute in the Class entity class, even though it’s in the corresponding table. Step 3 of the mapping algorithm would add it to that table.

d) Look at the MyMusic.accdb example we’ve been using. Here’s how it might be “reverse-engineered” to an ERD (harder):

Run this ERD through steps 1, 1.1, and 3 of the algorithm. Notice that the result is precisely the RDB of MyMusic.mdb.
Step 3 of the mapping algorithm mentions 1:1 relationship classes. Let's take an example:

According to step 3, we could either add deptID as a new column to the Employees table, or we could add empID as a new column to the Departments table. It turns out the second idea is the right one:

Employees(empID, empName)

Departments(deptID, deptName, empID)

Here, the “empID” column in Departments tells us who the manager of the department is. **Can you draw a conclusion here that would help us refine step 3 to recommend which primary key should be added to the “other side” as a foreign key?**