8.7 Homework: ER Diagrams & DB Schemas

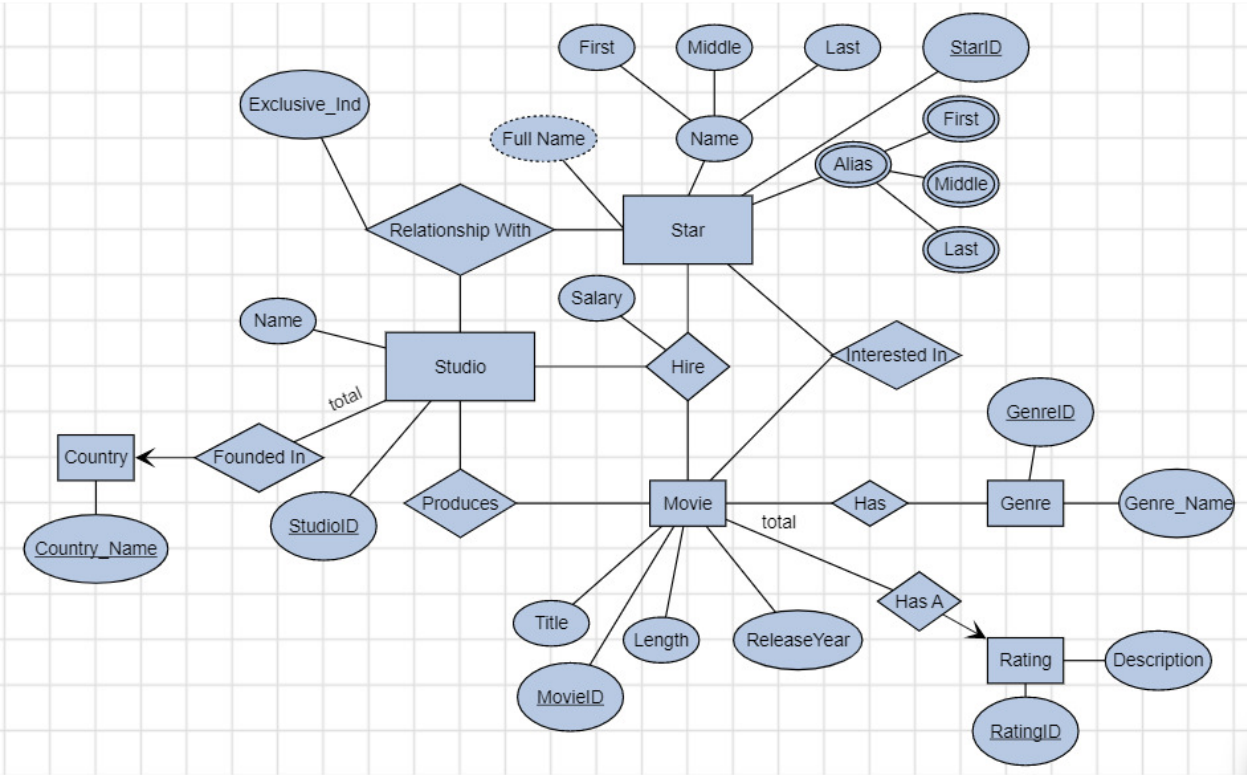
H. Diana McSpadden (hdm5s)

1. [30%] Create an ER diagram for the following database scenario.

Make sure to include an indication of the cardinality of relationships and indicate any mandatory relationships (total participation). State any assumptions that you make!

Simple IMDB Database

- Movie stars obviously have names (real and screen in some cases).
- Movie stars star in movies.
- Movies have titles, the year they came out, a genre, a length, and a rating.
- A rating can be G, PG, PG-13, or R.
- Movies are produced by studios.
- There are various different studios around the world.
- Movie studios hire actors to be in their movies and pay them a great deal of money (salary).



Movies Schema Assumptions

1. I realize I included relationships not described in the assignment details (e.g. Interest between a Star and a Movie), and this could be interpreted as over-engineering. I hope for grading purposes you understand that I was having fun and learning from the assignment. Thank you!
2. Movies exist prior to a Studio starting to produce them and prior to Stars being associated with them.

3. Stars and Movies can be interested in each other prior to being in a contract between the Producer/Star/Movie. This can capture the IMBD information about movies that are “in development.”
4. Stars can be in relationships with Studios prior to, and outside of being in a contract for a Movie. Some Stars are in an exclusive relationship with one Studio, but not all Stars are, so this is handled by a relationship attribute instead of cardinality.
5. A Movie can have many Genres.
6. Every Movie has at least one Genre.
7. A Movie can exist before it has been given a Rating.
8. Several Producers can work together to Produce a Movie.
9. The ID “StudioMovieProducesID” is the unique identifier for the relationship “Produces” between Studio and Movie. This ID is a required attribute of the “Hire” relationship between Studio, Star, and Movie. The “StudioMovieProducesID” can be referenced in multiple Hire relationships, but is unique to one Produces relationship. Some of this functionality must be enforced in the business logic layer (application code and/or stored procedures).
10. I do not have enough information in the description to determine if a Star could be under multiple contracts for a single Movie with multiple Producers, but this model will support that functional requirement if it needs to be supported.

2. [10%] Convert Task 1's ER diagram to a table.

Star (StarID, First_Name, Middle_Name, Last_Name)

Alias (StarID, Alias_First, Alias_Middle, Alias_Last)

Country (Country_Name)

Rating (RatingID, Description)

Studio (StudioID, Name, Country_Name)

Movie (MovieID, Title, Length, ReleaseYear, RatingID)

Genre (GenreID, Genre_Name)

Studio_Star (StudioID, StarID, Exclusive_Ind)

Studio_Movie_Star_Hired (StudioID, StarID, MovieID, Salary)

Studio_Movie_Produces (StudioID, MovieID)

Movie_Genre (MovieID, GenreID)

Movie_Star_Interest (StarID, MovieID)

Database Considerations

1. I named tables that represent relationship using the ENTITY-A_ENTITY-B notation.
2. In the **Movie** table, the RatingID will need to allow NULLS until a movie has been completed and rated by the Motion Picture Association, OR we could create an “Unrated” rating in the **Rating** table.
3. For the relationship table that tracks interest between a movie star and movie, I added the word “Interest” to the table name to be more clear about what the table is representing.
4. For the relationship table that tracks that a studio is producing a movie, I added the word “Produces” to the table name to be more clear about what the table is representing.

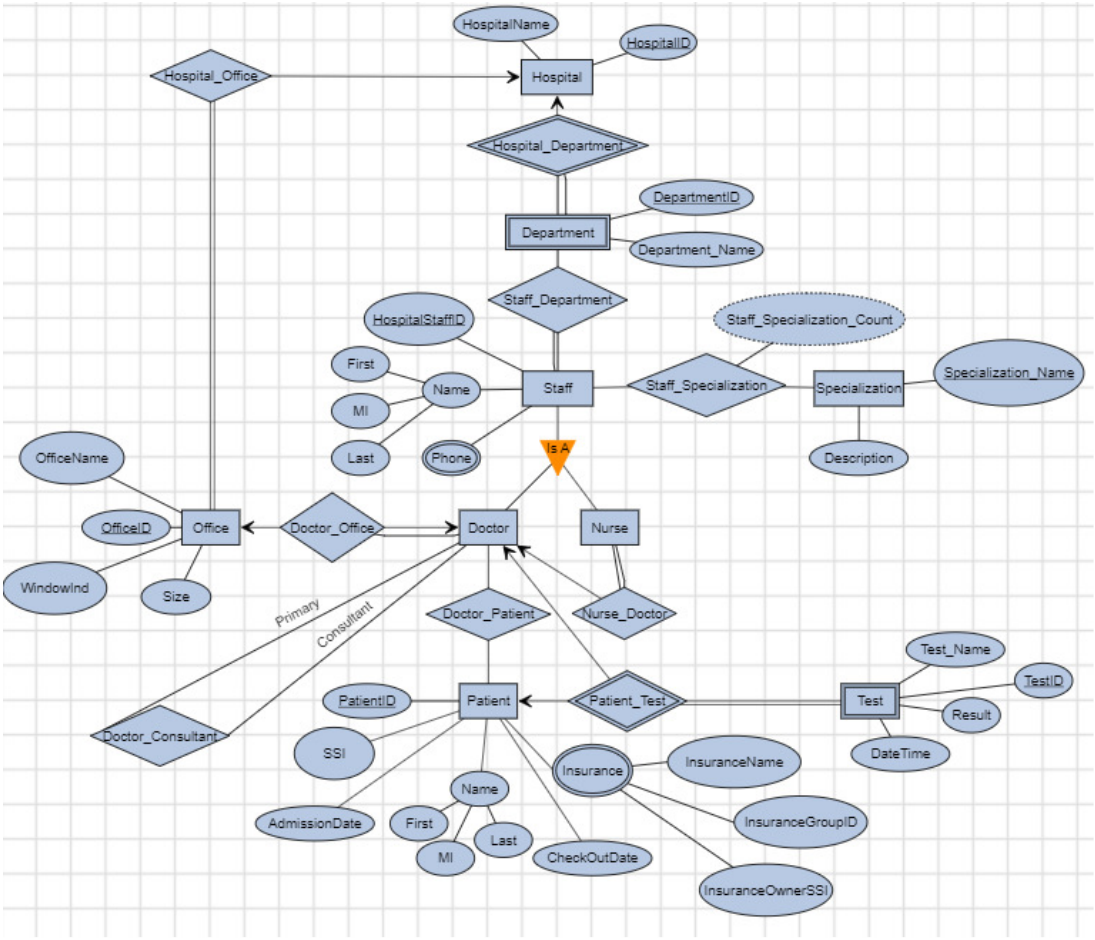
5. For the relationship table that tracks the “hired for” relationship between Studio, Movie and Star, I added the word “Hired” to the table name to be more clear about what the table is representing.

3. [40%] Create an ER diagram for the following database scenario.

Make sure to include an indication of the cardinality of relationships and indicate any mandatory relationships (total participation). State any assumptions that you make!

Hospital Database

- Hospital staff consists of doctors and nurses.
- All hospital staff have a hospital ID, name (first name, middle initial, and last name), and phone number.
- Doctors have an office and up to three specializations.
- Doctors have nurses that work with/assist them.
- Nurses have one specialization and belong to a specific department in the hospital.
- All nurses work with doctors (no exception). Many nurses can work with one doctor.
- Doctors sometimes consult with a colleague (who is another doctor).
- Doctors treat patients.
- Patient information is collected by the hospital and include SS# (social security number), name, insurance, date of admission, and date checked out.
- A patient can undergo a number of tests.
- Doctors perform tests.
- Tests have a unique ID, name, a result, and a date and time the test was performed.



Hospital Database Assumptions:

1. Staff can only work for one hospital. This model will not work to track staff across multiple hospitals.
    a. HospitalStaffID must be unique across all hospitals using the database.
2. I decided that Doctors also needed to be assigned a Department, and that all staff are assigned at least one Department.
    a. Nurses having a functional requirement that they have a single department will need to be enforced by code in the software or stored procedures. Alternatively, the Department relationship could be between only Nurses and Doctors could have a relationship with the Hospital, instead of Department, but I did not create that in this model. In my model, Doctors can be in a relationship with multiple Departments, so I believe it will be flexible enough to not restrict and flexible enough to provide greater reporting functionality (i.e. report the Departments a Doctor works for/with).
3. Each department is for only one hospital. As written, this model cannot work to track similar departments across hospitals; however, if that functionality is required, then DepartmentType can be created as a reference table, and each Department can be identified with a DepartmentType. We could then compare same department types across multiple hospitals.
4. It would be possible in this system, by SSI, to track a Patient across multiple hospitals. However, the HIPPA and other security concerns are extremely important to address in the database due to health records, and using SSI.
5. All doctors have an office: as is implied by the "total" annotation on the "Doctor_Office" relationship.
6. Because Office Numbers (i.e. OfficeName) may be duplicated across hospitals, I added a separate OfficeID, unique to the database, is used as the index for the table.
7. To the office table, I added an indicator as to whether the office has a window, and a size attribute just because it seemed interesting.
8. The functional requirements for doctors having no more than three specializations, and nurses having one specialization will need to be handled in the Business Logic/Tier and/or programmatically in stored procedures. The count of specializations per Staff person can be inferred from the relation table, as is noted in the schema.
9. The Doctor who orders a Test for a Patient is stored in the Patient_Test relationship as the Ordering_DoctorID attribute. This attribute is required (both the database and the Business Logic tier enforce).
10. Patients can have multiple insurers.
11. All Departments are in a Hospital.
12. All Staff are in a Department.
13. All Tests are for a Patient and do not exist without an associated Patient.

4. [20%] Convert Task 3's ER diagram to a table.

Hospital (HospitalID, HospitalName)

Office (Office_ID,OfficeName, HospitalID, WindowInd, Size)

Department (DepartmentID, HospitalID, Department_Name)

Staff (HospitalStaffID, First_Name, MI_Name, Last_Name)

Staff_Phone (HospitalStaffID, Phone_Number, Phone_Type)

Staff_Department (HospitalStaffID, DepartmentID)

Specialization (Specialization_Name, Description)

Staff_Specialization (HospitalStaffID, Specialization_Name)

Doctor (HospitalStaffID, OfficeID)

Nurse (HospitalStaffID, Doctor_ HospitalStaffID)

Doctor_Consultant (Primary_ HospitalStaffID, Consultant_ HospitalStaffID)

Patient (PatientID, SSI, AdmissionDate, CheckOutDate, First_Name, MI_Name, Last_Name)

Patient_Insurance (PatientID, InsuranceGroupID, InsuranceName, InsuranceOwnerSSI)

Doctor_Patient (Doctor_HospitalStaffID, PatientID)

Test (TestID, PatientID, Doctor_HospitalStaffID,Test_Name, Result, DateTime)

Database Considerations

1. I added a Phone_Type attribute to the Staff_Phone table to support functionality to save cell vs. home vs. work vs. etc phone types. There is a decision as to whether have the Phone_Type attribute have a Phone_Type table to reference or if the text “CELL”, “HOME”, ‘WORK” can be used and enforced in business layer code or stored procedures.
2. There is an unfortunate situation that can occur with this model. When a Doctor is no longer active at a Hospital, but the inactive Doctor has Nurses assigned to them. Since Nurses must be assigned a Doctor, we do not want NULL values in Nurse.Doctor_HospitalStaffID. One solution is to create a placeholder Doctor whose ID can be associated with Nurses who otherwise will not have a Doctor_HospitalStaffID, or create an “Active_Ind” on the Doctor record (this would be the preferred method for referential integrity purposes), and then have a business process that we check for Nurses assigned to inactive Doctors and fix the situation via process instead of enforcing it in the database.
3. In both the Doctor_Patient, and Test tables I refer to the HospitalStaffID as Doctor_HospitalStaffID for clarity.