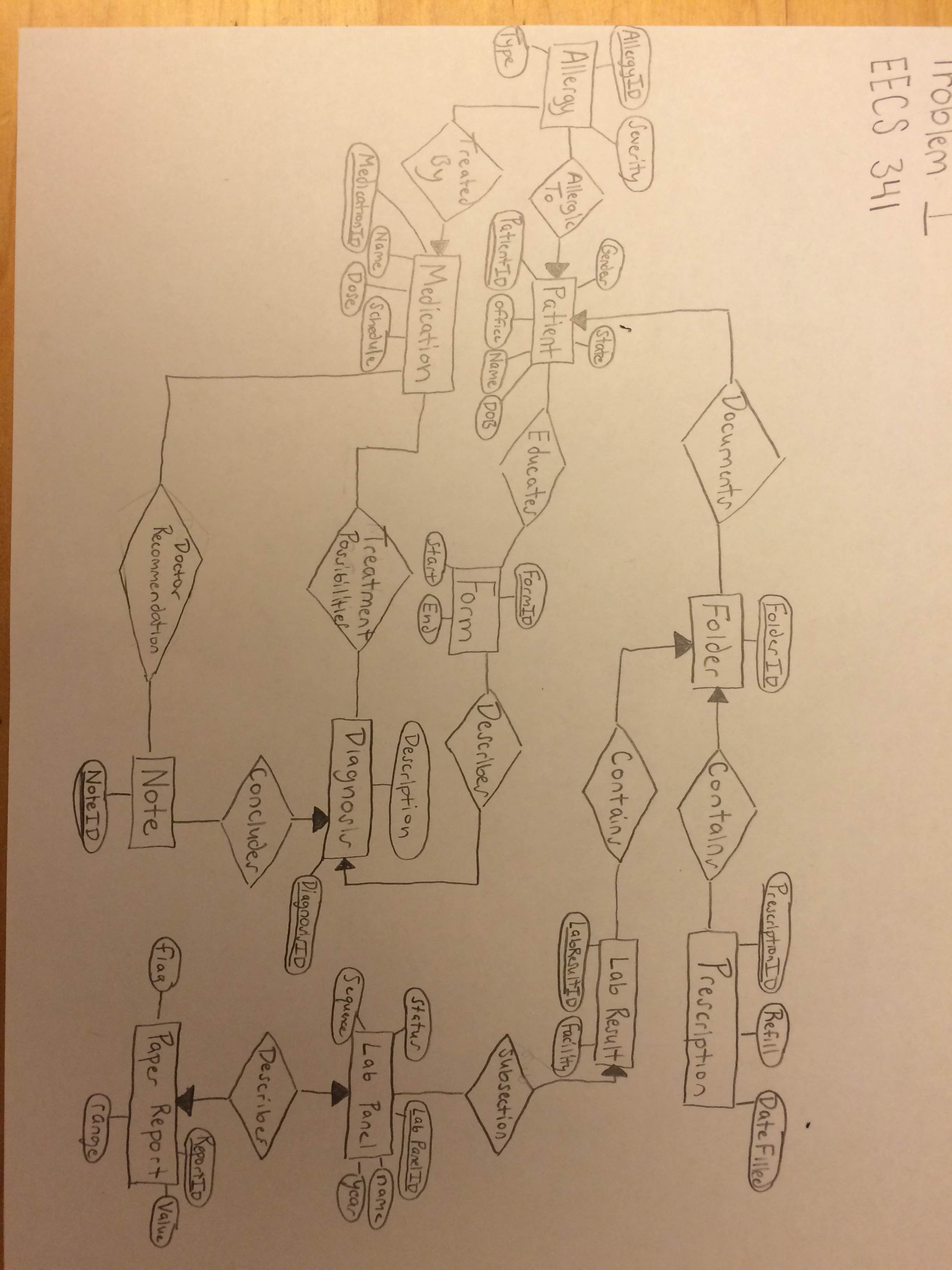
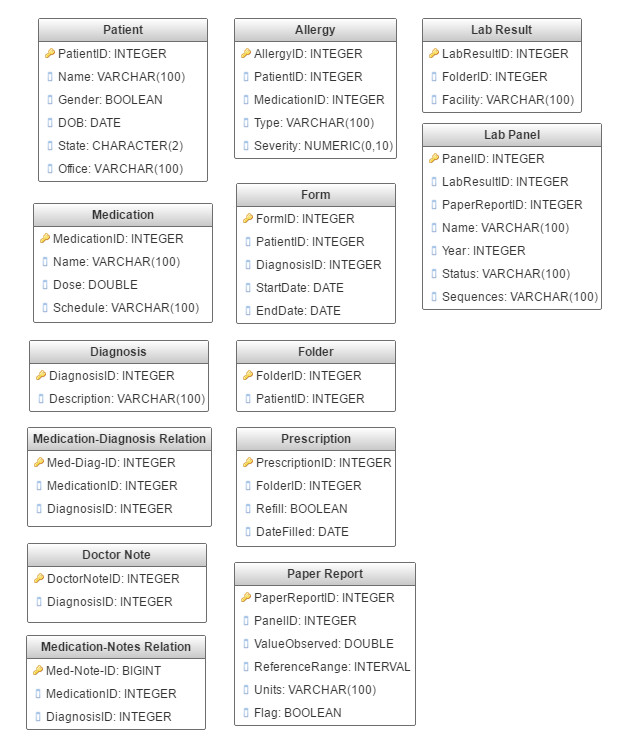
Jacob Alspaw

Intro Databases

23 September 2016

**1**



\*\*\*I forgot to label the cardinality constraints for problem 1 so I will list them below. Cardinality constraints restrain the relation between two entities. For each of the two entity types you can specify a pair of indices like (1:N). The left character may be 0 or 1, while the right character may be 1, N, or M. The first index is a lower bound on the relationship’s cardinality, while the second index is a sort of upper bound.

Cardinality constraints for problem 1:

* Each patient will have at least one folder regardless of whether or not there identity is known. (1:N)
* Each patient could be allergic to many substances or not at all. (0:N)
* One medication can treat many allergies, but some may have no treatment. (0:N)
* A patient may have many forms, but also none. (0:N)
* A folder may or may not contain a prescription. (0:N)
* A folder may or may not contain a lab result. (0:N)
* A lab result must have a lab panel. (1:N)
* A doctor’s note may or may not recommend a medication. (0:N)
* A diagnosis may or may not be treated by a medication. (0:N)
* At least one form will always describe one diagnosis. (1:N)

For additional constraints, I would say some column tables may not accept null values. Certain information must be accessed by the hospital like lab results which always have some form of a result even if it’s inconclusive, it is still a result. Or patient name because the hospital must refer to them by something even if the patient’s true identity is known.

For problem 1, I inferred:

* One medication can treat many allergies.
* Medications-Diagnosis Relation is a many-to-many relationship.
* Medication-Note Relation is a many-to-many relationship.
* Doctor Notes and diagnosis is a many to one relationship.
* A patient can have more than one folder if the original gets full.
* Office and facility were not put in their own table, but they could be. It would be a new table for both entities with an ID column (Integer), and a name column (Varchar). The Office and Facility columns in the old Schema would then be changed to the new respective ID columns to match the new Tables.

2

**A.** Will be found on the next page.

**B.** As is, the design only allows for a one-to-one relationship. To keep all the transactions by a customer, we will need a one-to-many relationship between the contract table and the transaction table, such that one contract is aware of many transactions. The design would be changed to remove the arrow pointer over the transaction table and replace it with a straight line.

**C.** As is, the design allows for a many-to-many relationship between the contract and the product. The contract will define the products’ prices after negotiation and based on the company. If the price were to be the same for each customer, then the design would need to remove the relationship between the Product table and the Contracts table. The price column would be transferred onto the Product table where each entry would remain constant for each customer.

