Transforming ER diagrams to Relational Schema Diagrams

Our conceptual model gets expressed as an ER diagram, but our implementation will most probably be in a relational database, so we need to transform our ER diagram to a Relational Schema diagram.

Normally you would create at least one table per entity type, but you will need to create extra tables for multivalued attributes and many-to-many relationships.

Now, what are the rules for Relational Schema diagrams ? They are simple.

- Tables are represented by their name, and attributes. A pretty representation looks like this:

Employee

SSN	FirstName	MiddleName	LastName

Figure 1: A simple table

Whereas an ugly representation would look like this:

Employee(Ssn, FirstName, MiddleName, LastName)

- Identifier attributes (keys) should be underlined
- Foreign key references are represented by an arrow from the foreign key to the primary key.

To make reading easier, I would abbreviate Primary Key as PK and Foreign Key as FK.

1. Transforming entities

We will create a new table for each entity type (normally with the same name as the entity type). We will transform the attributes as follows:

- **Simple** attributes become simple attributes on the table.
- The **identifier** gets marked on the table by underlining, same as in the ER diagram
- For **composite** attributes, we add to the table only the atomic pieces.
- **derived** attributes in the ER diagram do NOT appear on the table (since they wouldn't be stored)
- For **multivalued** attributes we need to create a new table; this new table has as attributes the primary key of the entity type (as a foreign key reference to the original entity type) and the multivalued attribute. The primary key of this new table is usually composed of ALL the attributes (including the primary key of the entity type).

As an example, the following ER diagram

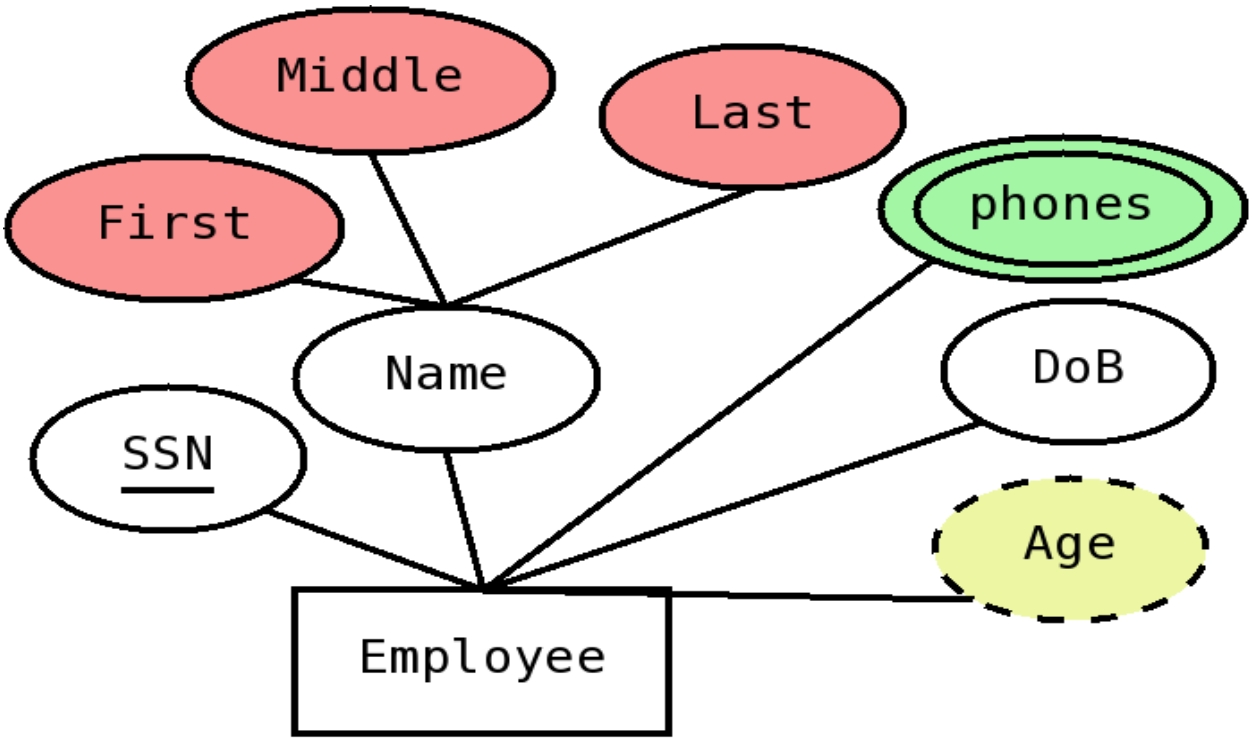


Figure 2: ER diagram for an Employee entity

Would get transformed to the following Relational Schema Diagram:

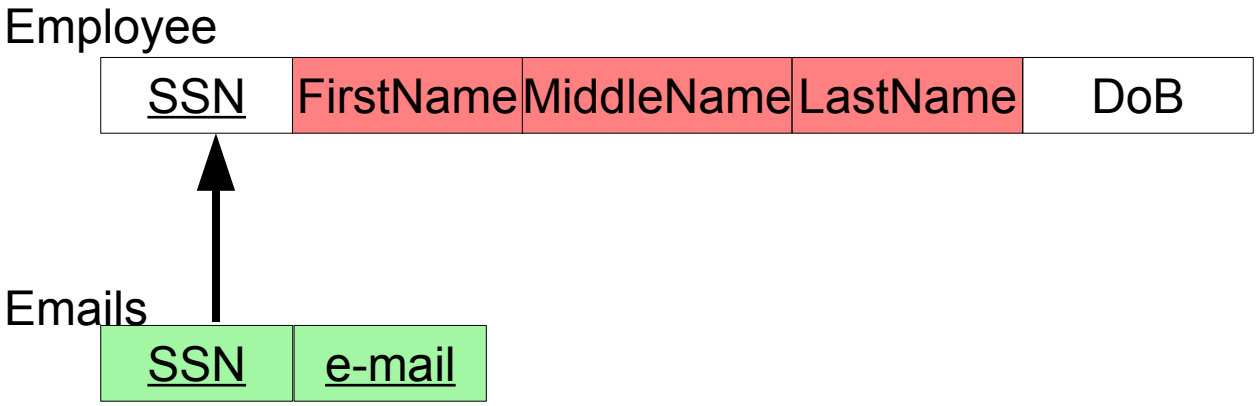


Figure 3: Relational schema diagram corresponding to the ER diagram in Figure 2

- The two simple attributes (ssn and DoB) get mapped directly, with SSN underlined since it is the identifier (primary key).
- The composite ER attribute Name does NOT appear on the table, but instead we have only its **atomic pieces**, first, middle and last, with a suitable naming convention (so they become FirstName, MiddleName and LastName)
- The derived attribute, Age, does NOT appear on the able (since it wouldn't be stored)
- For the multivalued attribute, emails, we need to create a new table. We call this table Emails (another good name would be something as Employee-Emails). This table has the PK of the Employee table (the original entity) as a FK reference to the Employee table, and a field email,

which would contain one value of the multivalued atribute (the email). The PK of this table is composite, and contains both attributes.

2. Transforming relationships

Relationships associate two entities, and in the relational model, associations are made through foreign keys (FKs); so we would need to put foreign keys in our relational schema. The transformation to be made depends on the cardinality of the relationship; we will first cover how to transform one-to-one and one-to-many relationships.

If we have a one-to-many relationship, between two entities, say A and B, where an A is related to many Bs, but a B can only be related to one A, we can put a reference to A on the table corresponding to B. If there are any attributes attached to the relationship, they would go on the same table.

For example, the following ER diagram

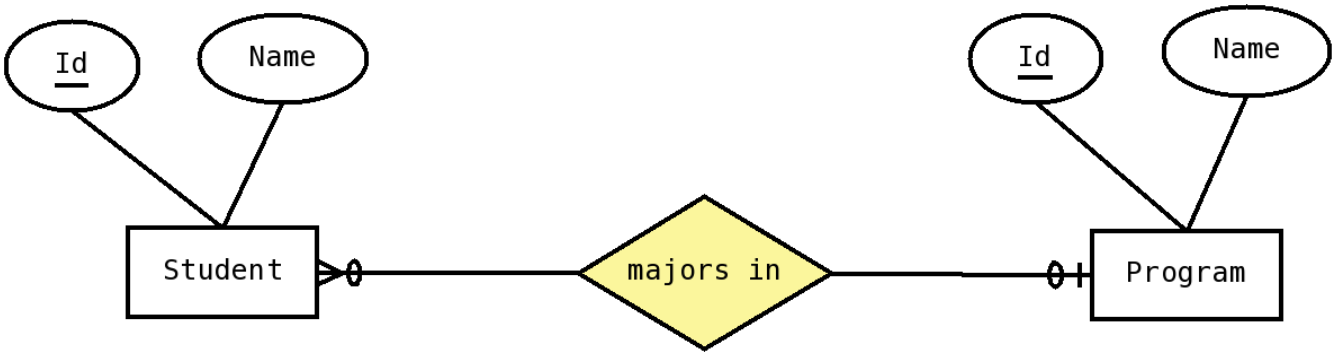


Figure 4: ER diagram with one-to-many relationship

Would be transformed to the following relational schema diagram:

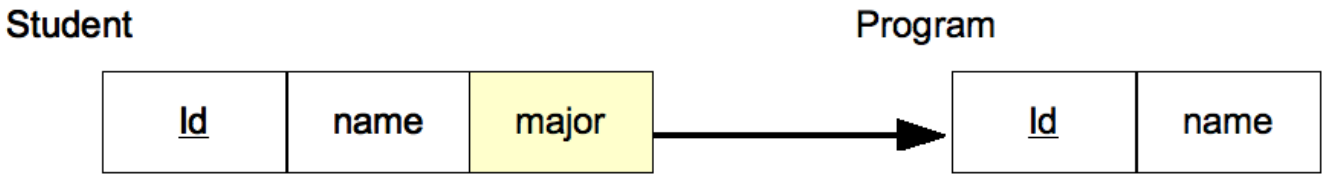


Figure 5: Relational schema diagram for one-to-many relationship

For one-to-one relationships, we can choose which side to put the reference on (since either would work), we normally choose the table for which we expect to have less rows to put the reference on.

Now, for many-to-many relationships, we cannot put the reference in either table, since there is the potential to have many references per row in either side (which would be a multivalued attribute); the solution is to create a new table, that includes references to the corresponding entities. For example, the following ER diagram:

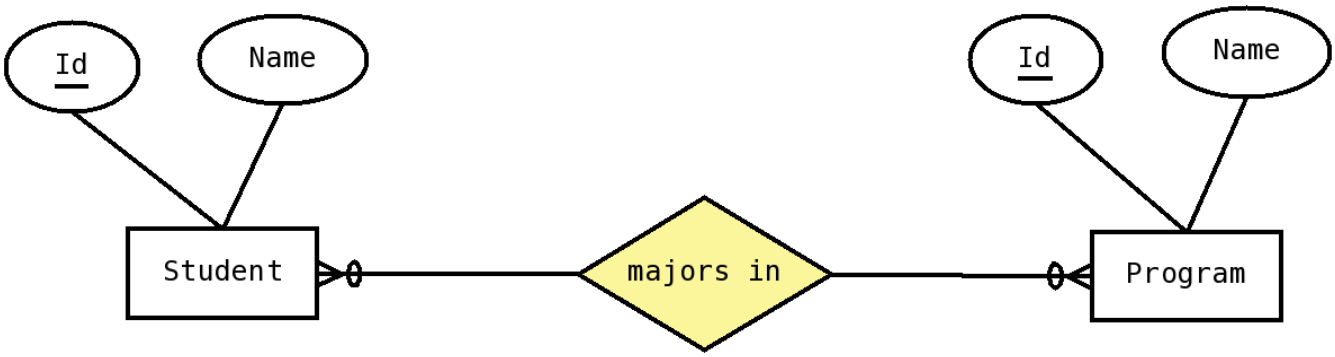


Figure 6: ER diagram with a many-to-many relationship
would be transformed into the following relational schema diagram:

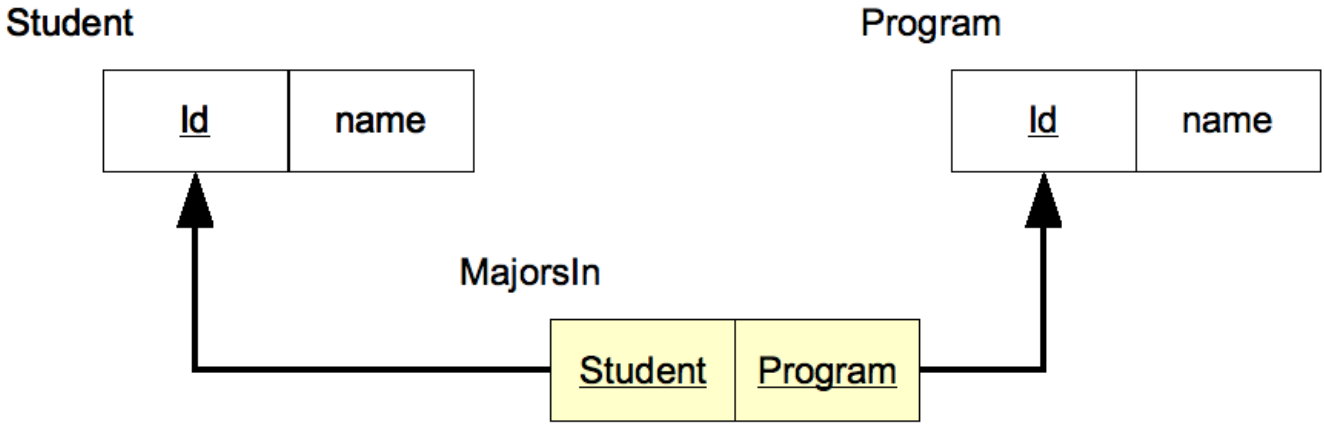


Figure 7: Relational Schema diagram corresponding to an ER diagram with a many-to-many relationship

If there are attributes attached to the relationship, then we put those attributes in this new table.

3. Recursive Relationships

Recursive relationships are dealt like any other relationship, except that the foreign key references will be to the *same* table; in the case of a one-to-many relationship we will have a FK reference to the same table, and in a many-to-many relationship we will have *two* fields, each one being an FK reference to the same table. For example, the ER diagram in figure 8, would be transformed to yield the relational schema in figure 9.

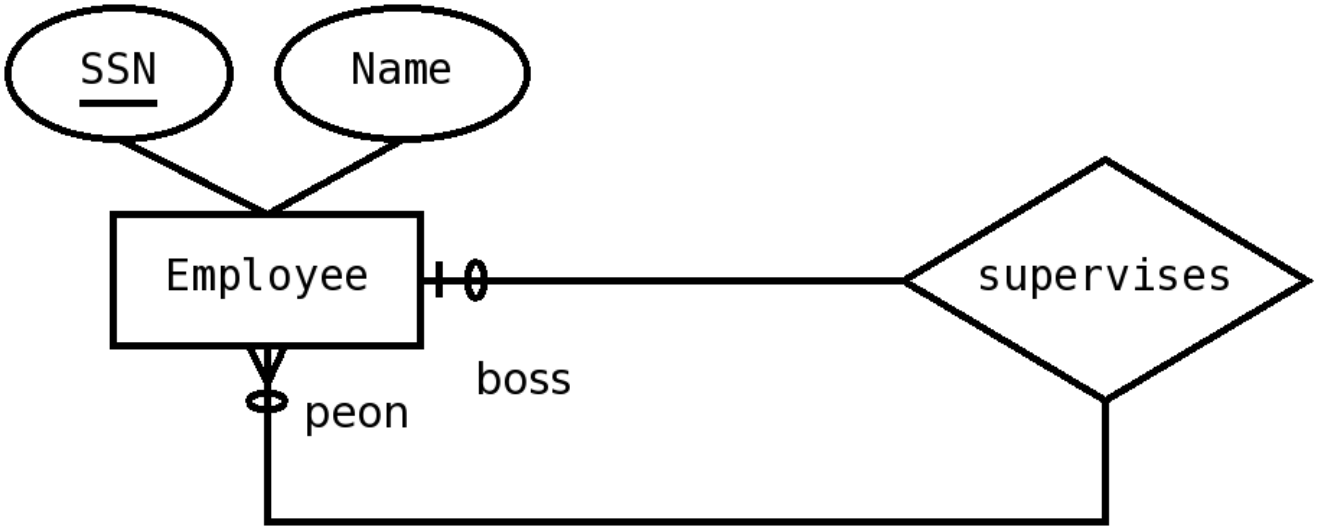


Figure 8: ER diagram with a one-to-many recursive relationship

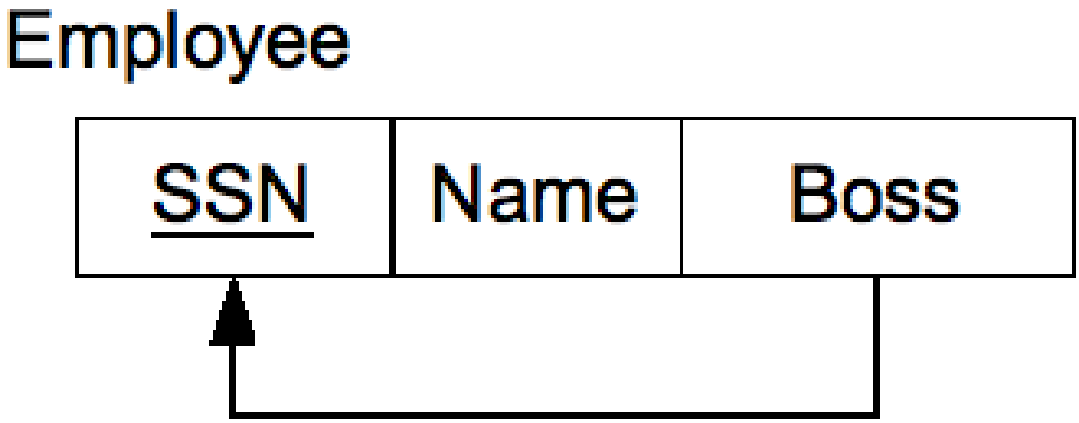
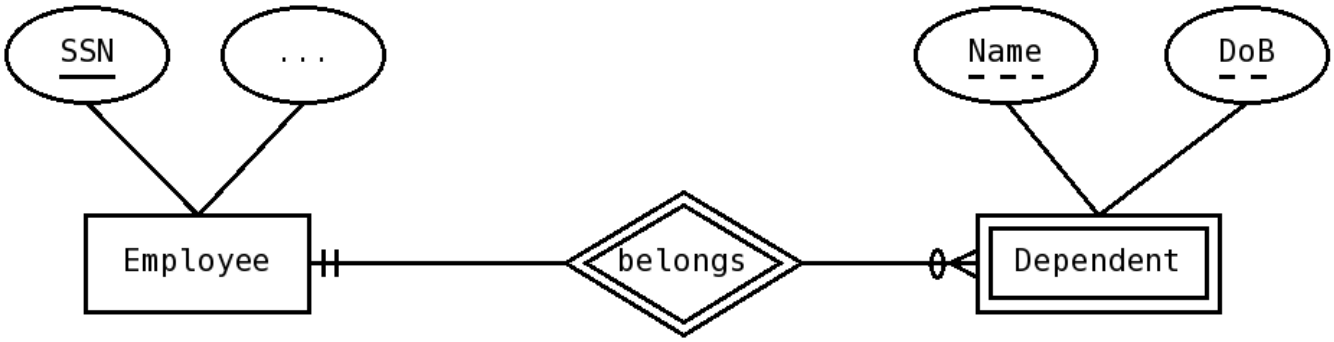


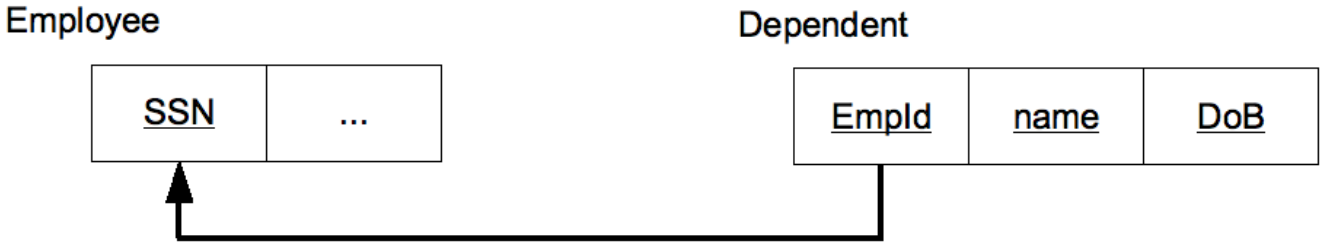
Figure 9: Relational schema diagram for a recursive, one-to-many relationship (supervises)

4. Weak entities

Weak entities are those that have no identifier, and need the strong entity to be fully identified. They always have a one-to-many relationship with a strong entity, called the *identifying relationship*. They are transformed similarly to one-to-many relationships, except that the identifier of the strong entity becomes part of the primary key for the weak entity. Notice the weak entity would already have a reference to the strong entity, due to the transformation of the one-to-many relationship; the only change is that this reference is now part of the primary key for the weak entity's table. As an example, the following ER diagram:

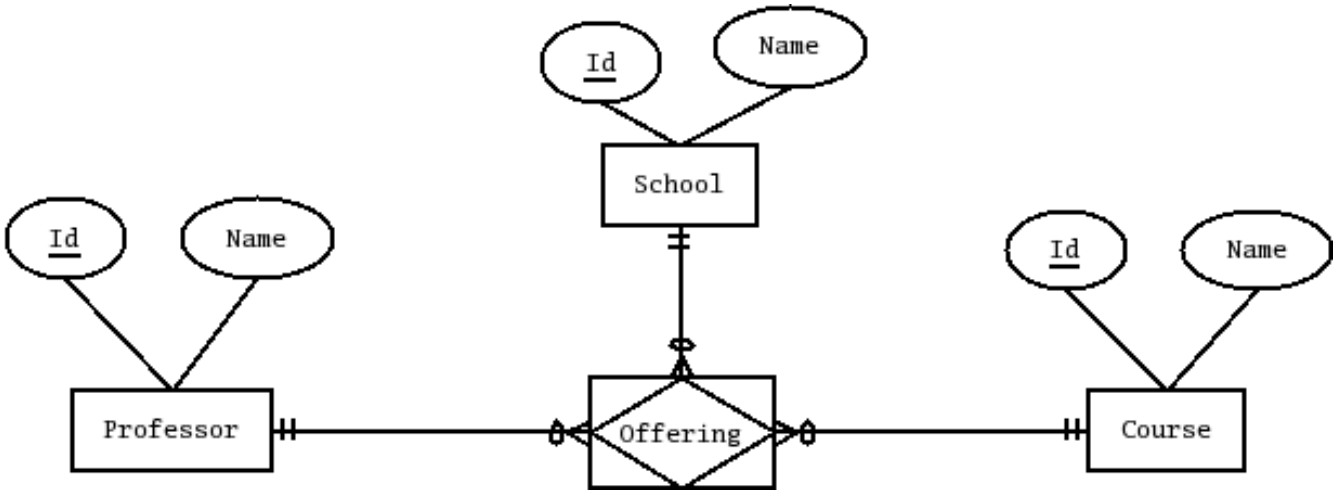


Would transform into the following relational schema diagram:

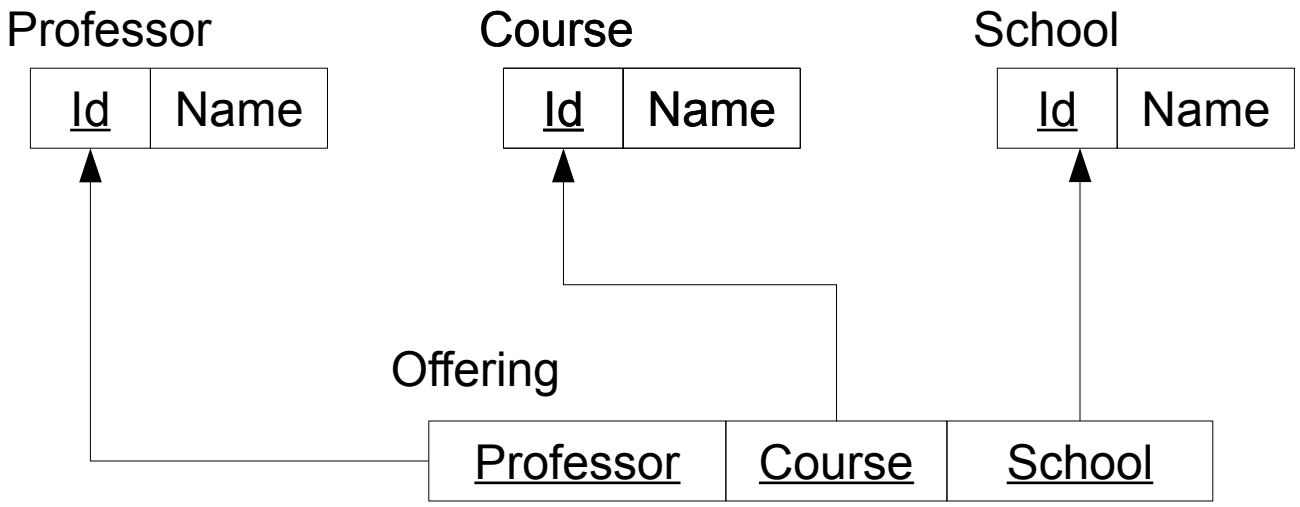


5. Associative entities

For an associative entity, we create a new table, with the references to the associated entities conforming the primary key. For example:



would transform into:



6. Comparing the models

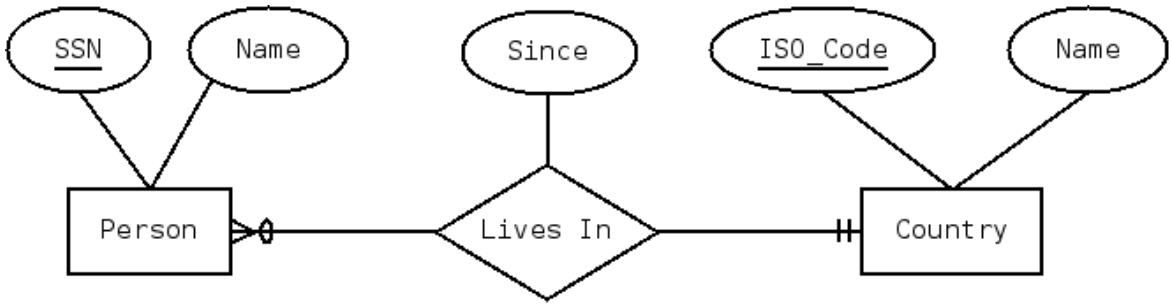
We can represent the same data in both the relational and the ER model; however, there will be several differences:

- The ER model has three kinds of things; Entities, Relationships and Attributes, and it uses relationships to associate entities
- The Relational model has only one kind of thing; a table; both entities and relationships may be represented as tables (or as parts of tables); it uses foreign keys to associate *rows* from different tables (you can also think of a table as associating attributes)
- Entities in the ER model have an identity independent of the values of its attributes; in the relational model rows contain only values. The identity of the row is its value of the attributes in the primary key.

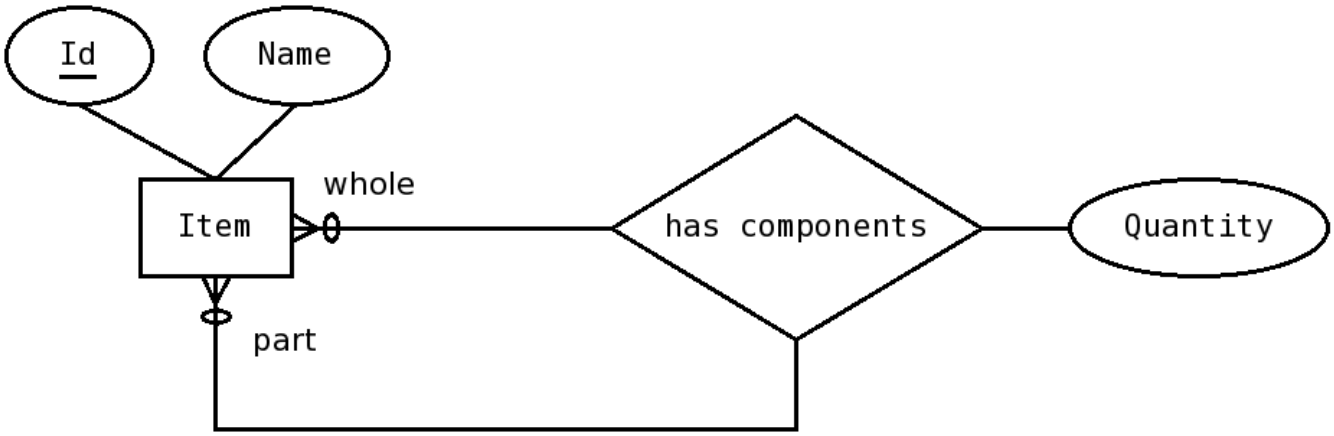
7. Exercises

(starred ones have solutions provided)

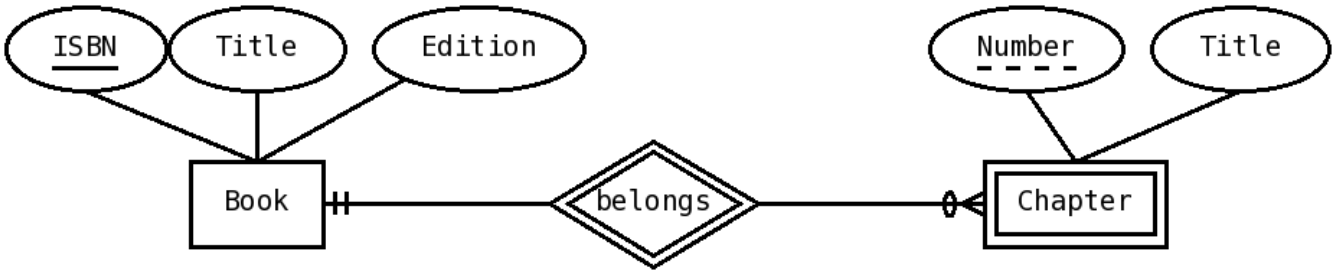
1. Transform the following ER diagram into a relational schema diagram.



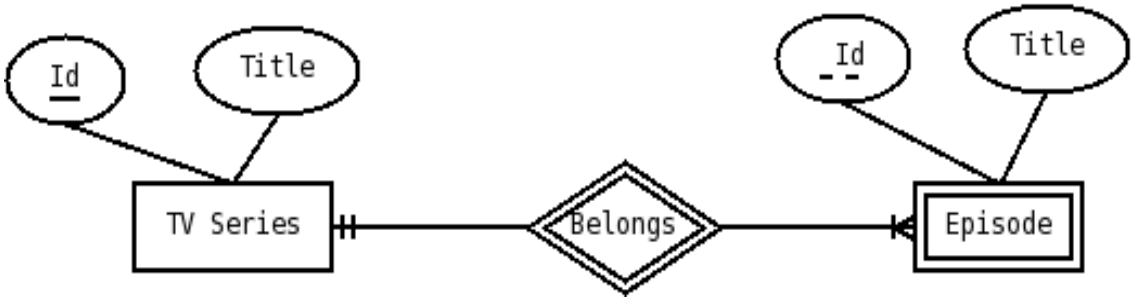
2. Transform the following ER diagram into a relational schema diagram.



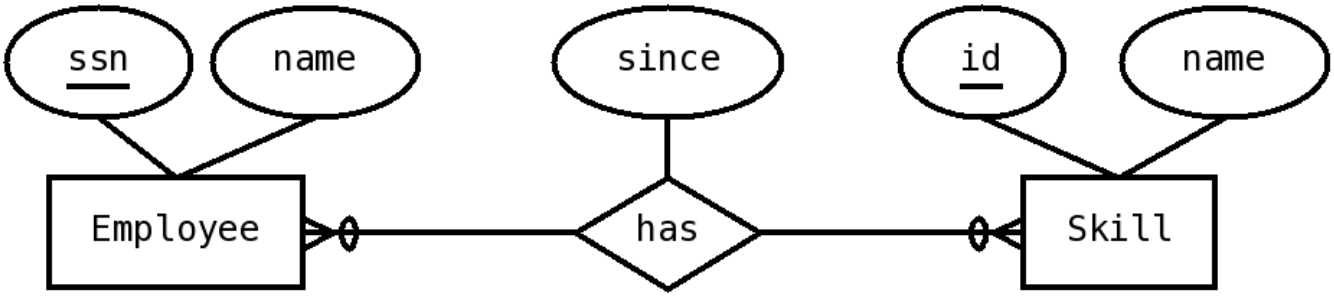
3. Transform the following ER diagram into a relational schema diagram.



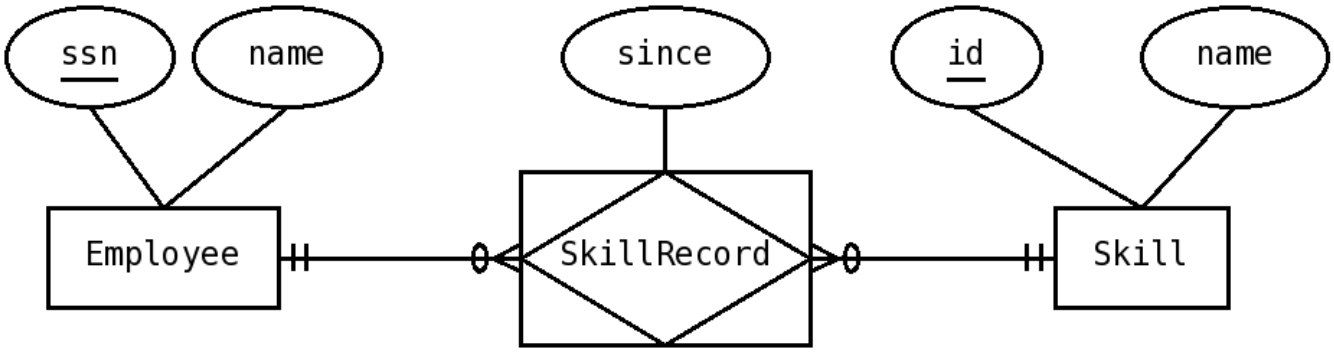
4. Transform the following ER diagram into a relational schema diagram.



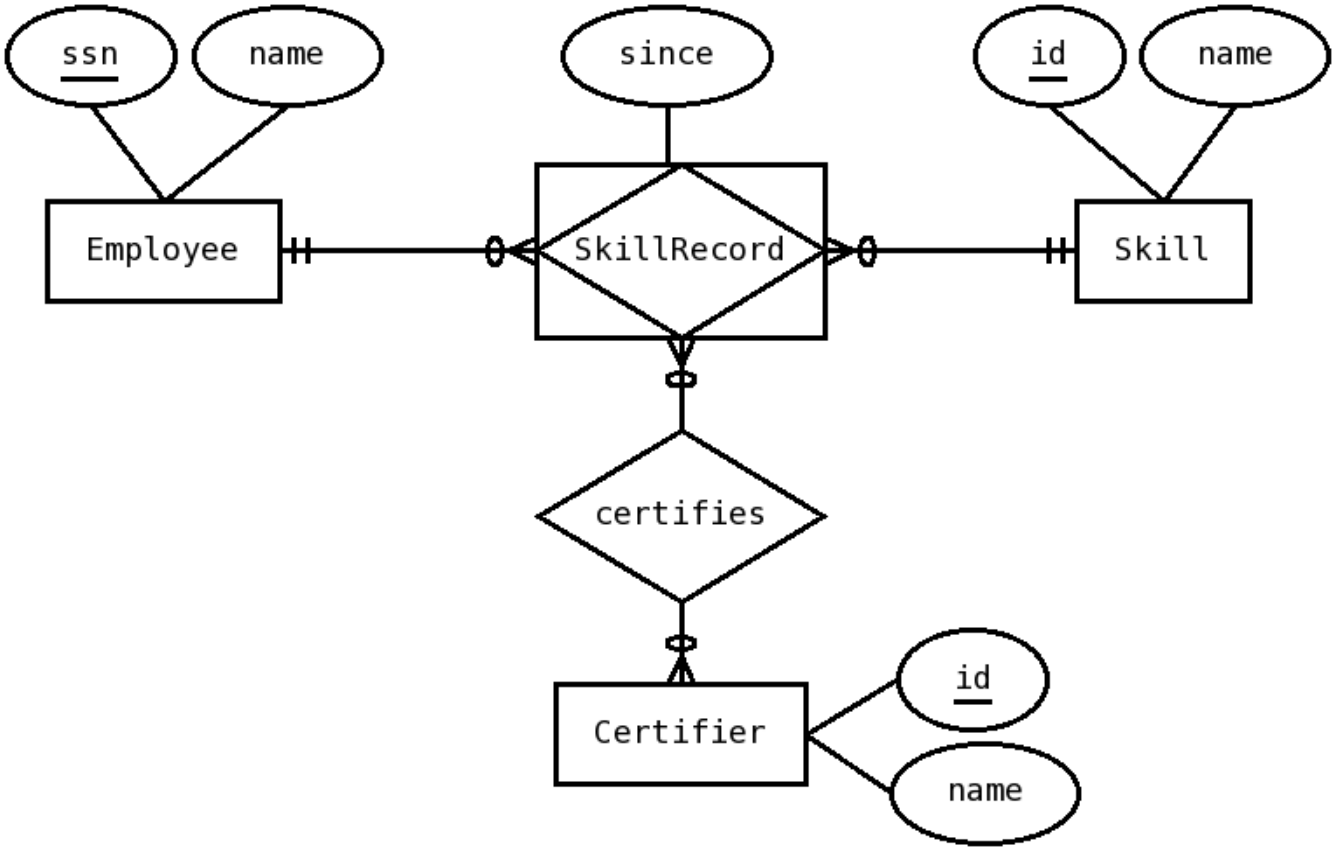
5. Transform the following ER diagram into a relational schema diagram.



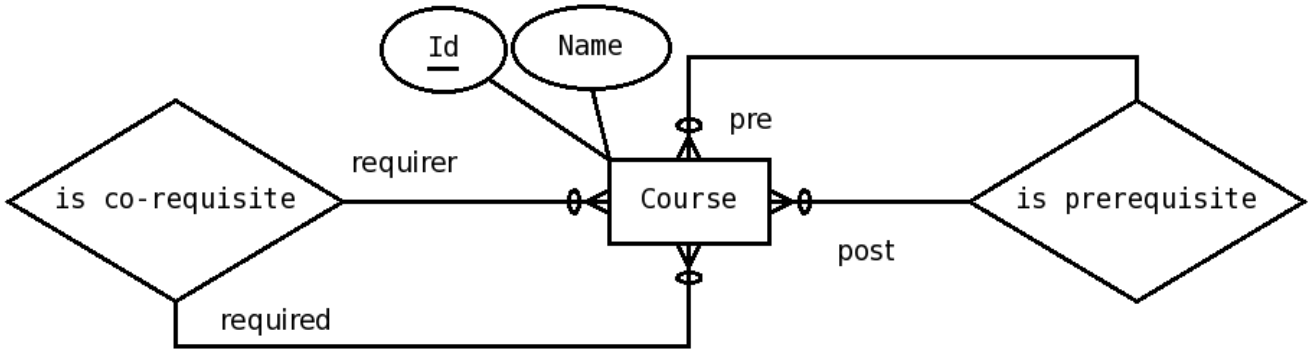
6. Transform the following ER diagram into a relational schema diagram.



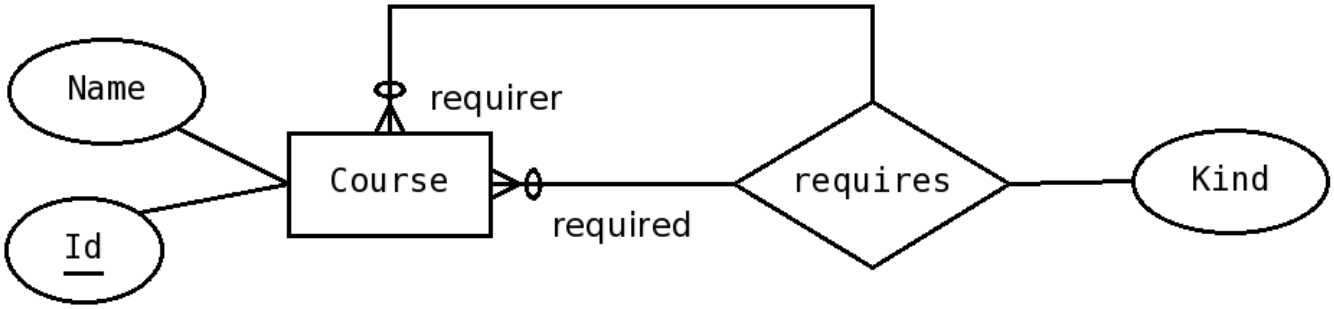
7. Transform the following ER diagram into a relational schema diagram.



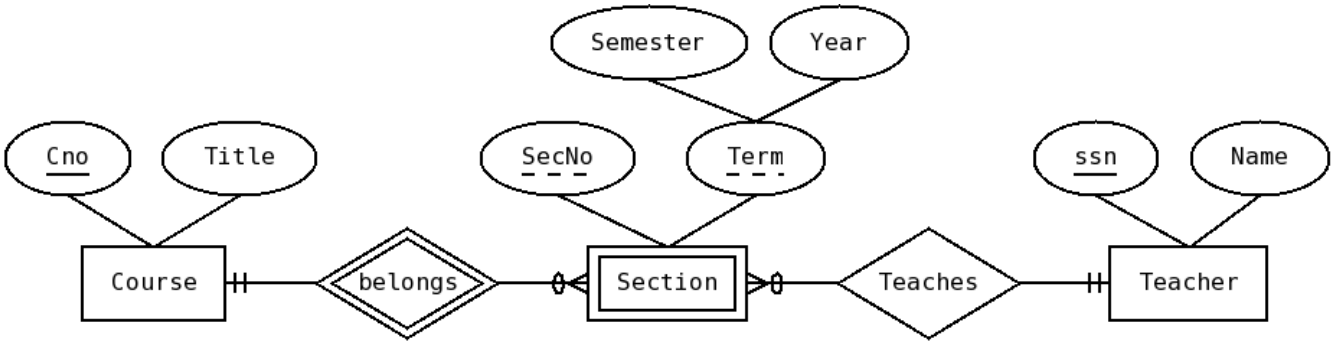
8. Transform the following ER diagram into a relational schema diagram.



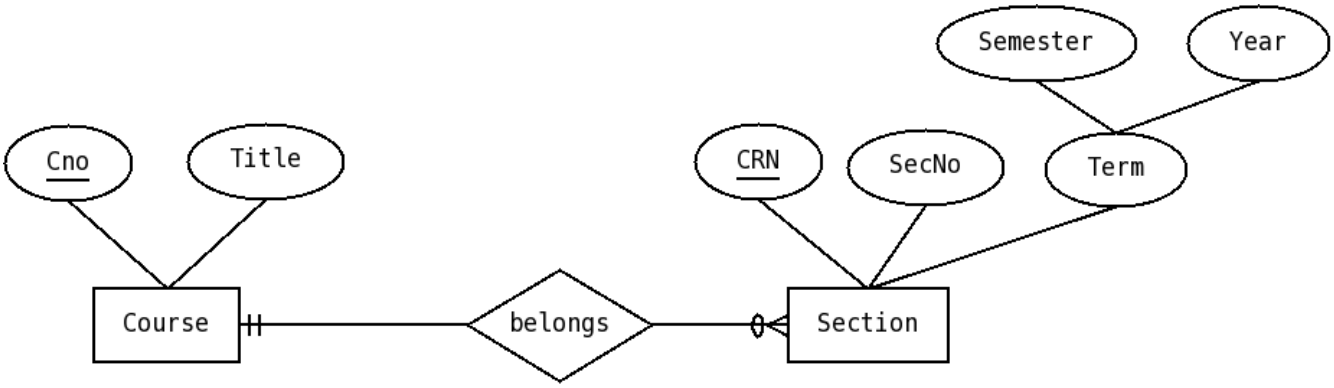
9. Transform the following ER diagram into a relational schema diagram.



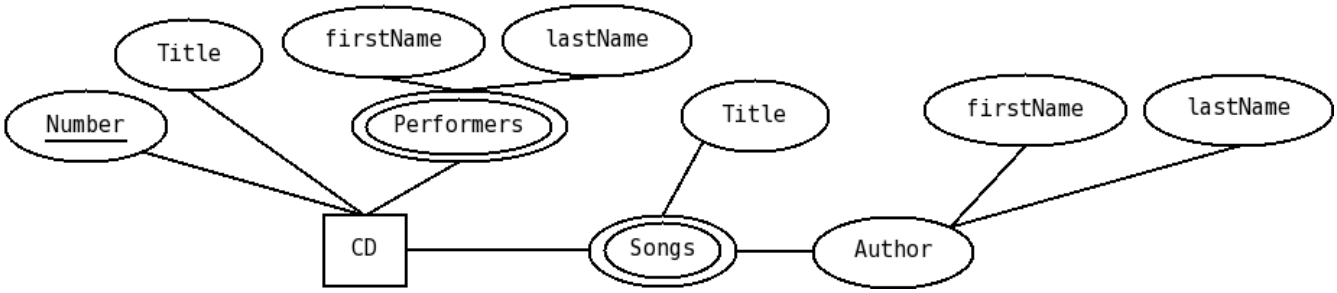
10. Transform the following ER diagram into a relational schema diagram.



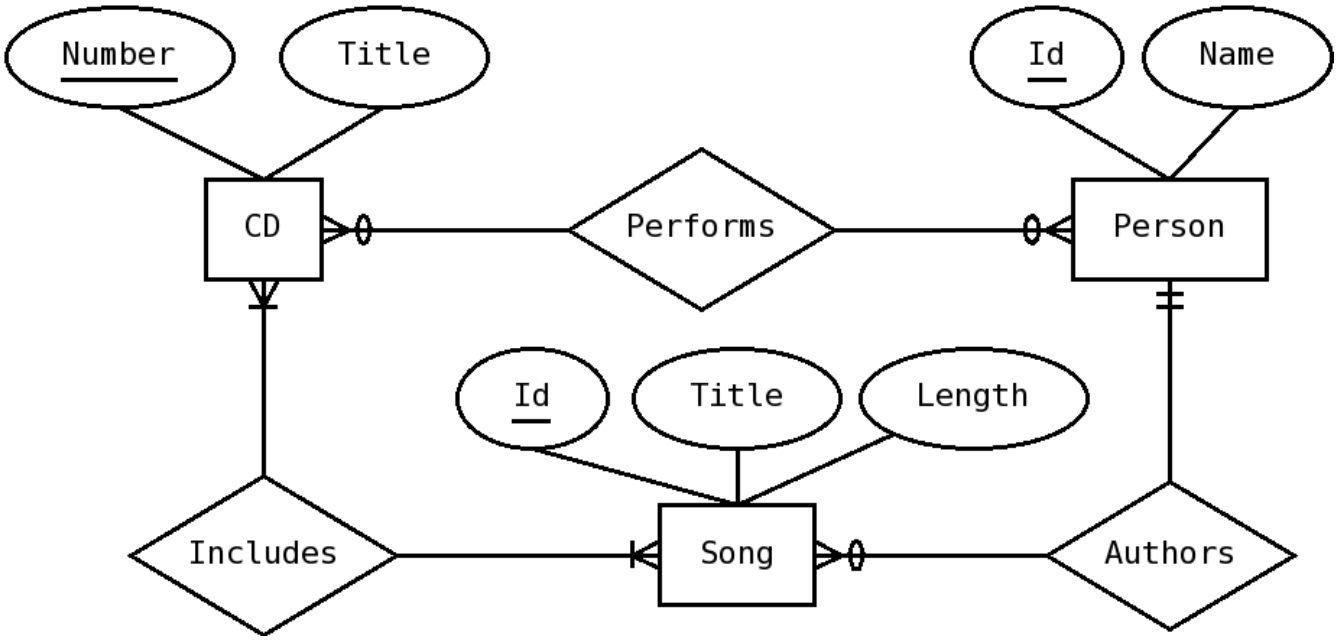
11. Transform the following ER diagram into a relational schema diagram.



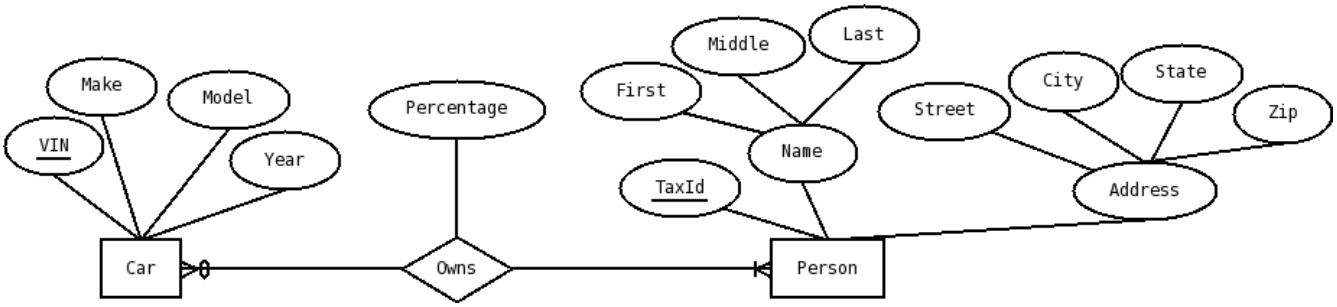
12. Transform the following ER diagram into a relational schema diagram.



13. Transform the following ER diagram into a relational schema diagram.



14. Transform the following ER diagram into a relational schema diagram.



15. Transform the following ER diagram into a relational schema diagram.