Converting ER Diagram to Relational Database

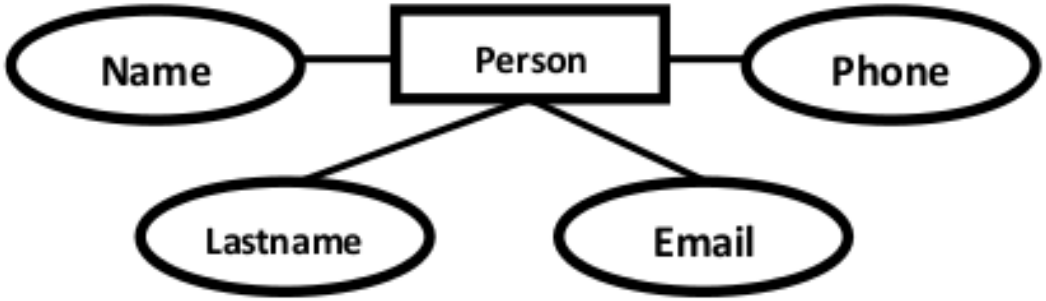
The ER diagram represents the conceptual level of database design meanwhile the relational schema is the logical level for the database design. We will be following the simple rules:

1. Entities and Simple Attributes:

- An entity type within ER diagram is turned into a table. You may preferably keep the same name for the entity or give it a sensible name but avoid DBMS reserved words as well as avoid the use of special characters.
- Each attribute turns into a column (attribute) in the table.
- The key attribute of the entity is the primary key of the table which is usually underlined.

It is highly recommended that every table should start with its primary key attribute conventionally named as TablenameID.

Taking the following simple ER diagram:



The initial relational schema is expressed in the following format writing the table names with the attributes list inside a parenthesis as shown below for

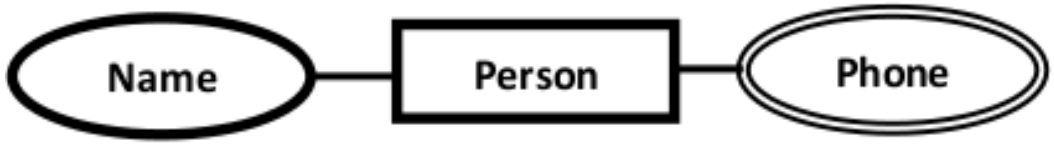
Persons (personid , name, lastname, email)

Persons is a Table. name, lastname, … are Table Columns (Attributes).

personid is the primary key for the table: Person

2. Multi-Valued Attributes

A multi-valued attribute is usually represented with a double-line oval.



If you have a multi-valued attribute, take the attribute and turn it into a new entity or table of its own. Then make a **1: N relationship** between the new entity and the existing one.

In simple words:

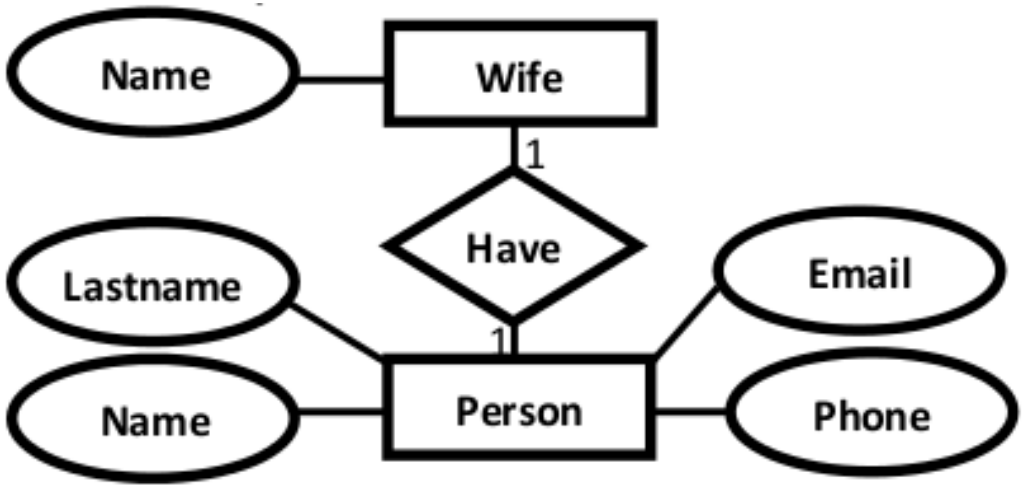
1. Create a table for the attribute.
2. Add the primary (id) column of the parent entity as a foreign key within the new table as shown below:

Persons (personid , name, lastname, email)

Phones (phoneid , ***personid***, phone)

personid within the table Phones is a foreign key referring to the personid of Persons

3. 1:1 Relationships



To keep it simple and even for better performances at data retrieval, I would personally recommend using attributes to represent such relationship. For instance, let us consider the case where the Person has or optionally has one wife. You can place the primary key of the wife within the table of the Persons which we call in this case Foreign key as shown below.

Persons (personid , name, lastname, email , ***wifeid***)

Wife (wifeid , name)

Or vice versa to put the **personid** as a foreign key within the Wife table as shown below:

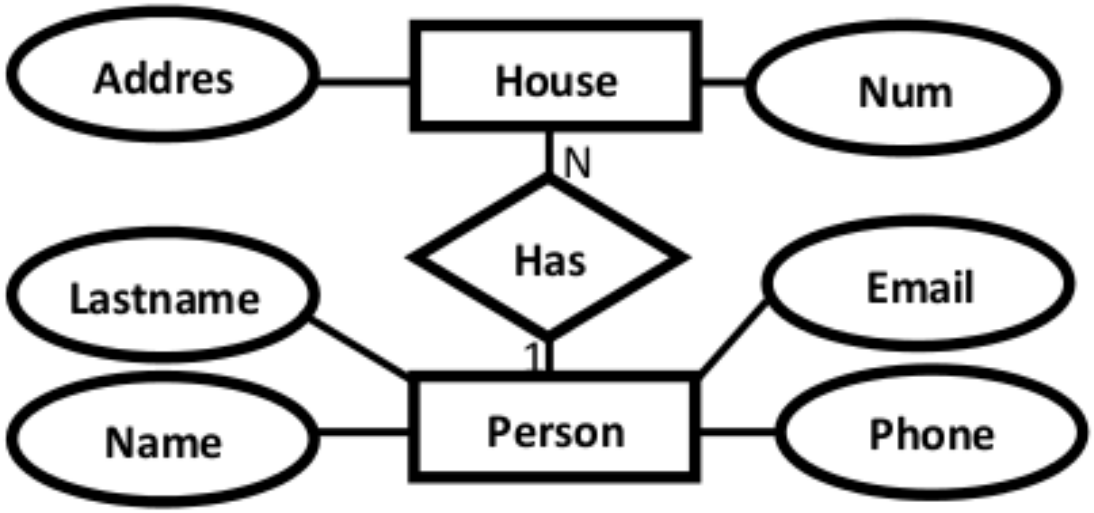
Persons (personid , name, lastname, email)

Wife (wifeid , name , ***personid***)

For cases when the Person is not married i.e. has no wifeID, the attribute can set to NULL

4. 1:N Relationships

This is the tricky part ! For simplicity, use attributes in the same way as 1:1 relationship but we have only one choice as opposed to two choices. For instance, the Person can have a **House** from zero to many , but a **House** can have only one **Person**. To represent such relationship the **personid** as the Parent node must be placed within the Child table as a foreign key but **not the other way around as shown next:**



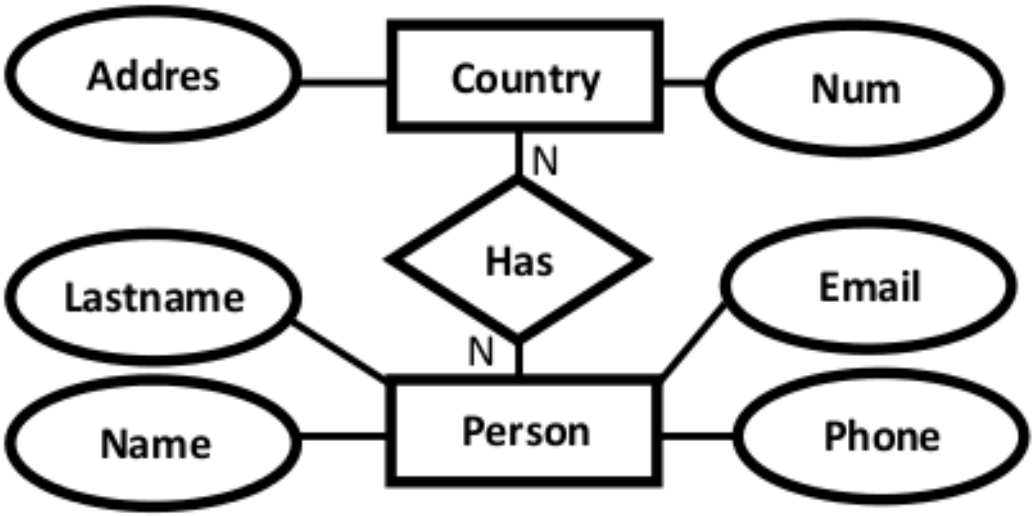
It should convert to :

Persons (<u>personid</u> , name, lastname, email)

House (<u>houseid</u> , num , address, ***personid***)

5. N:N Relationships

We normally use tables to express such type of relationship. This is the same for N − ary relationship of ER diagrams. For instance, The Person can live or work in many countries. Also, a country can have many people. To express this relationship within a relational schema we use a separate table as shown below:



It should convert into:

Persons (personid , name, lastname, email)

Countries (countryid , name, code)

HasRelat (hasrelatid , **personid , countryid**)

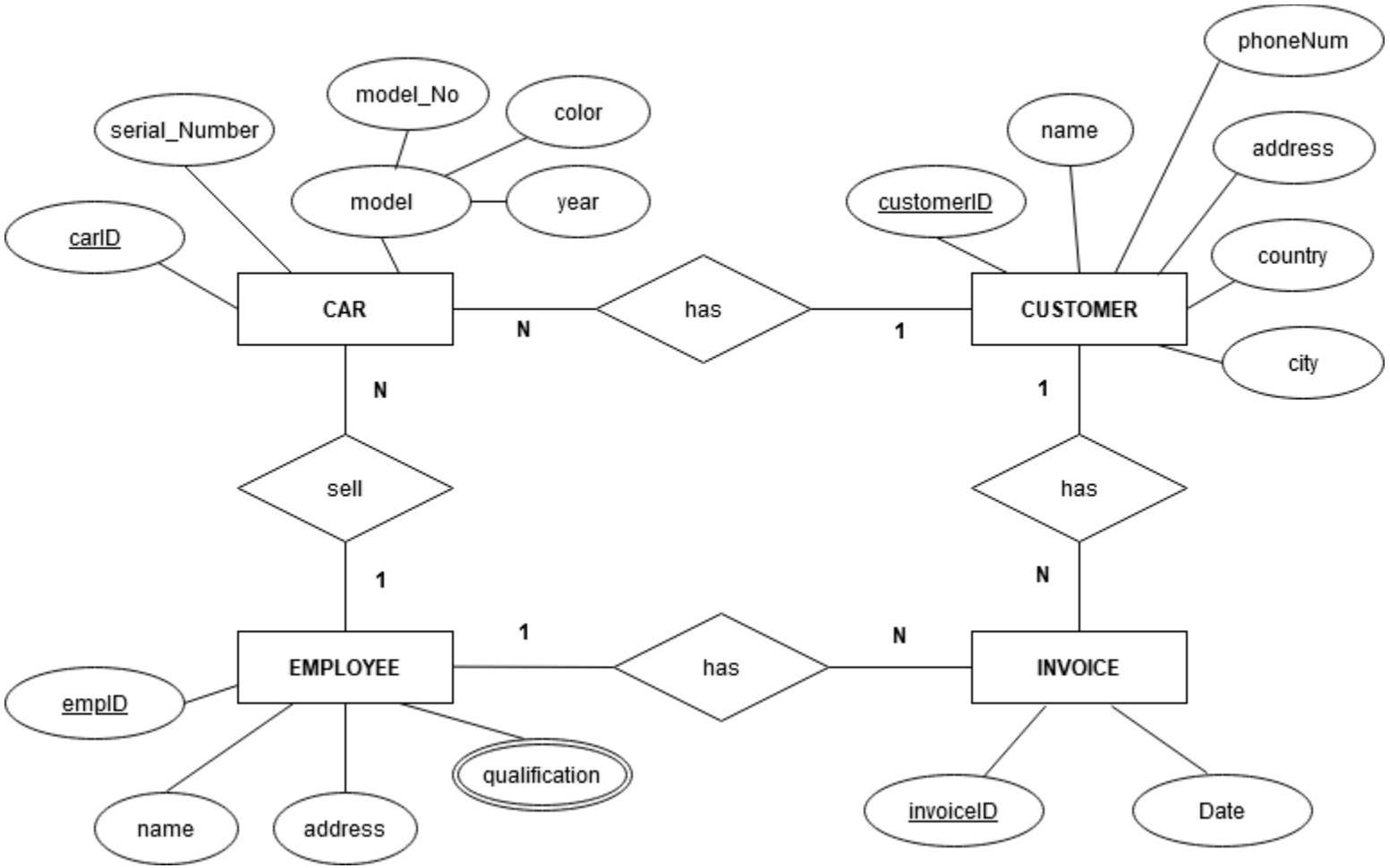
Relationship with attributes:

It is recommended to use table to represent them to keep the design tidy and clean **regardless of the cardinality** of the relationship.

Case Study

For the sake of simplicity, we will be producing the relational schema for the following ER diagram:

Transforming ERD to Relational Schema



Solution:

CAR (**carID**, serial_Number, model_No, color, year, customerID, empID)

CUSTOMER (**customerID**, name, phoneNum, address, country, city)

EMPLOYEE (**empID**, name, address)

EMPLOYEE-QUALIFICATION (**empID**, qualification)

INVOICE (**invoiceID**, date, empID, customerID)