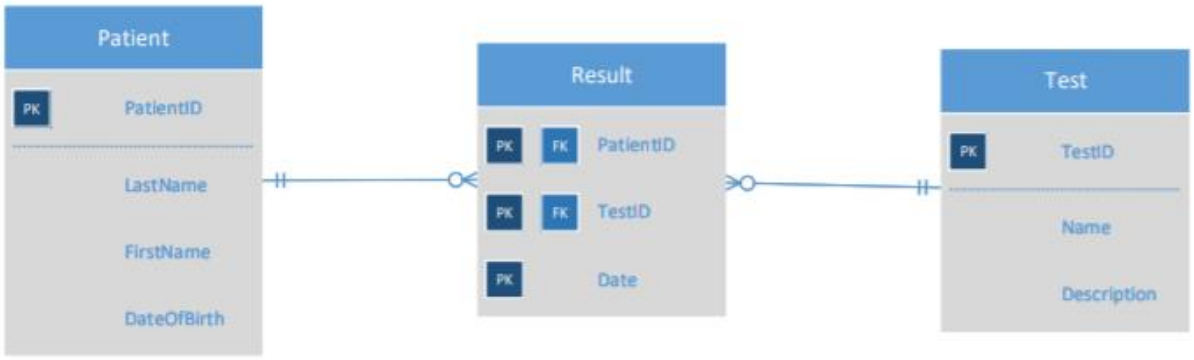
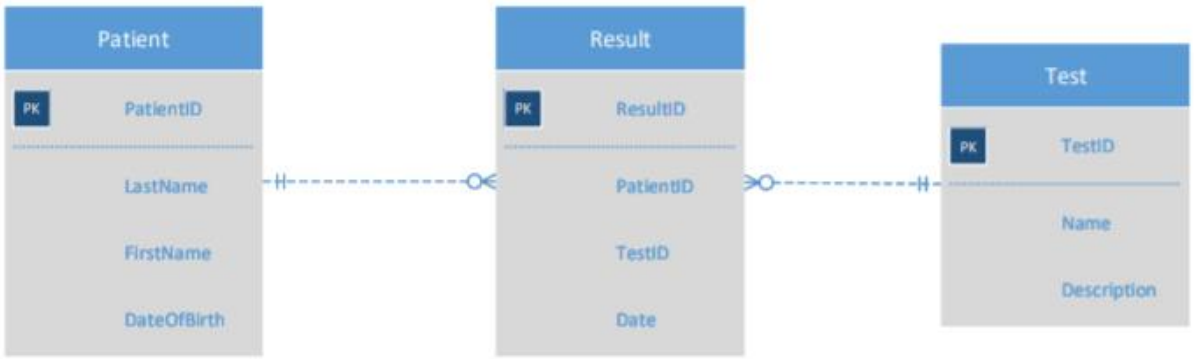
Part 2



The above design has two identifying relationships for Result. When we change the design to the model below, what else do we need to do to ensure the business rules can still be maintained? Please elaborate.



I have considered both the figures as figure 1 and figure 2 and when we change the design to figure 2 what we need to do to ensure that the business rules are still maintained:

- **So, here I initially explained with figure 1**
- **Secondly, explained about the figure 2**

FIGURE 1: (Identifying Relationship)

- In the first diagram we are using two identifying relationships among the entities to form a relationship between the three entities **Patient, Result, Test**
- As there is a many to many relationship between Patient and Test we use a **Associative entity** which is named as **Result** using the primary keys of the other entities
- Here we are using **Natural key** this attribute has a business meaning. Here we have a Natural Key that is composite so the Natural Key is inefficient as its very large there by **increases the overhead**
- In the first figure we see an Identifying relationship among each other, so we used a **solid line** and the **FK is contained in the primary key of the child entity**.
- So Result table is formed by taking the primary keys of parent entity i.e Result entity has PatientID and TestID among its primary keys used
- But because of the composite primary key making it too long we face **data Integrity issues** causing an effect on the accuracy and the consistency of data.

FIGURE 2: (Non-identifying Relationship)

- Here we use a **surrogate key** with a non-identifying relationship among them to form a relationship between the three entities **Patient, Result, Test**
- **Surrogate key** is an artificially produced value with no real world meaning often an integer
- **Natural Key is inefficient if it's very large as the overhead increases (Fig. 1) so to avoid this we use a surrogate key (Fig. 2)**
- As there is a many to many relationship between Patient and Test we use a **Associative entity** named as **Result** using the primary keys of the other entities but **that Foreign Key (FK) is not used as the Primary Key (PK) in the child entity**
- We use a **dotted line** to show the relationship between the entities
- We put the **original key attributes as mandatory** i.e **PatientID in the Result which will refer to the primary key in the PatientID in the Patient table and TestID in the result which will refer to the primary key in the TestID in Test table**.
- Whenever a tuple is entered in the Result entity table we ensure that PatientID and TestID are entered.
- We maintain a **referential integrity for the original key attributes** i.e whenever a original key attribute is entered we lookup at the value in the related parent entity to ensure it's a good value
- Here we maintain a referential integrity with **PatientID** and **TestID**
- So by doing this and not using composite primary key which is too long so we **can avoid data Integrity issues** which was causing an effect in the accuracy and the consistency of data.
- And thus ensures that the business rules are thereby maintained