BCNF – Explained to a 8 Year old

Your pizza can have exactly three topping types:

- one type of cheese
- one type of meat
- one type of vegetable

So we order two pizzas and choose the following toppings:

```
Pizza     Topping      Topping Type
--------  ----------   -------------
1         mozzarella   cheese
1         pepperoni    meat
1         olives       vegetable
2         mozzarella   meat
2         sausage      cheese
2         peppers      vegetable
```

Wait a second, mozzarella can't be both a cheese and a meat! And sausage isn't a cheese!

We need to prevent these sorts of mistakes, to make mozzarella *always* be cheese. We should use a separate table for this, so we write down that fact in only one place.

```
Pizza     Topping
--------  ----------
1         mozzarella
1         pepperoni
1         olives
2         mozzarella
2         sausage
2         peppers


Topping      Topping Type
----------   -------------
mozzarella   cheese
pepperoni    meat
olives       vegetable
sausage      meat
peppers      vegetable
```

That was the explanation that an 8 year-old might understand. Here is the more technical version.

BCNF acts differently from 3NF only when there are multiple overlapping candidate keys. The reason is that the functional dependency x -> y is of course true if y is a subset of x. So in any table that has only one candidate key and is in 3NF, it is already in BCNF because there is no column (either key or non-key) that is functionally dependent on anything besides that key.

Because each pizza must have exactly one of each topping type, we know that (Pizza, Topping Type) is a candidate key. We also know intuitively that a given topping cannot belong to different types simultaneously. So (Pizza, Topping) must be unique and therefore is also a candidate key. So we have two overlapping candidate keys.

I showed an anomaly where we marked mozarella as the wrong topping type. We know this is wrong, but the rule that makes it wrong is a dependency `Topping -> Topping Type` which is not a valid dependency for BCNF for this table. It's a dependency on something other than a whole candidate key.
So to solve this, we take Topping Type out of the Pizzas table and make it a non-key attribute in a Toppings table.