<u>Normal Forms Qualifying Criteria</u> (revisited)

Given:

- A relation schema R and set of FDs F on R.
  [Note: Key needs to be computed from given FDs]

Assumptions:

- All FDs in F are non-trivial and left (side) irreducible.
  Note: Checking get simplified if your FDs is canonical and minimal -
- Right side is singleton
- Left side irreducible
- No Transitive FDs

Let me begin with middle point 3NF and then take both sides BCNF and 2NF.

<u>3NF</u>:

Check (1): for every FD X → Y that holds on a relation, one of following is true –

- X is key, OR
- Y is prime attribute

Alternate Check (2)

- <u>All non-prime attributes are dependent on key</u>.
  Requires direct dependence (and not transitively) - if your FD set is minimal, there will be no transitive FDs, and therefore transitivity check is automatically taken care of.

How both definitions converge?

Check criteria (1) correctly maps to alternative criteria (2) -

- Y is prime attribute ==> exclude prime attributes, i.e. check only for non-prime attributes
- X is key ==> determinant is key

Even though, alternate definitions are less formal, and our formal approaches cover them well; approach still finds its application space.

Example: Consider relation Department (<u>DNo</u>, DName, DLocation, MgrSSN); a novice database designer may have chosen relation like this, and based on common sense she may have decided DNo as the key.

Given FDs are

DNO → DName
DNO → MgrSSN.

It should be noticeable that DLocation, which is non-prime attribute, is not determined by the key; therefore it is not in 3NF.

However our Check (1) will not be able to detect that it is not in 3NF? Why?

It is because DNO is not correct key of the relation Department for given set of FDs? {DNO, DLocation} is the Key. Let us apply both checks again by taking {DNO, DLocation} as key -

- Check (1): Now both FDs violate 3NF requirement.
- With alternate check (2): "all non-prime attributes should be determined by the key"; and both non-primed attributes are not determined by the key?
- Therefore from both perspectives, the relation is not in 3NF.

BCNF:

- For every FD X → Y that holds on a relation, X is key.
  Note that 3NF accommodates FDs in which X is not the key but Y is prime attribute. This exemption is no more in BCNF. This makes BCNF stricter.

BCNF criteria may also be specified as restrictor form of alternative definition of 3NF – "all ~~non-prime~~ **other** attributes should be determined by the key".
Note that the relation Department in example above, may pass through BCNF check as well?

2NF:

Check (1): for every FD X → Y, one of following is true –

- (a) X is key, OR
- (b) Y is prime attribute, OR
- (c) There is another FD A → X, where A is the key; i.e. Y is transitively determined by key.

Alternate Check (2):

- All non-prime attributes are dependent on key (transitive dependence is allowed).

How both definitions converge?

In check (1), (a) demands key to be determinant, (b) says exclude prime attributes, and (c) says that allow transitive dependence too. And this way check (1) turns out to be same as alternative check (2).

Example: Let us take the same used in 3NF discussion.
Consider relation Department (DNo, Dname, Dlocation, MgrSSN); and Dno has been taken as key; and FDs are –

DNO → Dname
DNO → MgrSSN.

Considering alternative definition, we discover that relation is not in 2NF as well.
But may slip through other definition?

1NF: attributes have "atomic" values

Determine Normal Form of a Relation (the process)

```
Input: Relation R, set of FDs F (assumption: all FDs in F are non-trivial and left (side)
irreducible)
Output: Highest Normal Form of a relation
Process:
    Compute key set K
    If there is any FD f that violates BCNF requirement THEN
        State "relation is not in BCNF, FD f violates"
        Also output other FDs that are violating BCNF requirement.
        If there is any FD that violates 3NF requirement THEN
            State "relation is not in 3NF, FD f violates"
            Also output other FDs that are violating 3NF requirement.
            If there is any FD that violates 2NF requirement THEN
                State "relation is not in 2NF, FD f violates"
                Also output other FDs that are violating 2NF requirement.
            Else
                Relation is in 2NF
        Else
            Relation is in 3NF
    Else
        Relation is in BCNF
```