Diagnosing BCNF is NP-hard in general.
Possible that no lossless join dependency preserving BCNF decomposition.

3NF: looks like BCNF but also allows $X \to Y \in F^+$ to imply that each attribute of Y is contained in a candidate key of R.

$R = \{A, B, C\}$ $\quad F = \{AB \to C, C \to B\}$

Minimal Cover

Definition
Two sets of dependencies F and G are equivalent iff $F^+ = G^+$.

Definition
A set of dependencies F is minimal if:
1) every RHS of a dependency in F is a single attribute
2) for no $X \to A$ is the set $F - \{X \to A\}$ equivalent to F.
3) for no $X \to A$ and Z a proper subset of X is the set $F = \{X \to A\} \cup \{Z \to A\}$

equivalent to F.

(2) efficiently check using
$A \in$ Compute $X^+(X, F - \{X \to A\})$

Theorem
For every set of dependencies F there is an equivalent minimal set of dependencies (minimal cover) → poly time

(3) $A \in$ Compute $X^+(Z, F)$

Finding Minimal Covers
loop:
1. replace $X \to YZ$ with $X \to Y$ and $X \to Z$
2. remove $X \to A$ from F if $A \in$ Compute X^+ $(X, F - \{X \to A\})$
3. remove A from LHS of $X \to B$ if
F if $B \in$ Compute $X^+(X - \{A\}, F)$
(we have a minimal cover here)
4. replace $X \to Y$ and $X \to Z$ in F by $X \to YZ$

A lossless-join 3NF decomposition that is dependency preserving can be efficiently computed.

```
func ComputeBCNF(R, F):
    result := ∅;
    F+ := min cover for F
    for each (X, Y) ∈ F+ do
        result := result ∪ {XY}
    if ¬∃ Ri ∈ result s.t.
(i)     Ri contains a candidate key for R then
(ii)        compute a candidate key K for R
            result := result ∪ {K}
    return result
```

line (ii): repeatedly remove attributes from R to turn superkey into candidate key using hill climbing

line (i): check if superkey using $R \stackrel{?}{=} \text{Closure}^+(R_i, F)$

$R = \{A, B, C, D\}$ $\quad F = \{A \to B, \ldots \to D\}$

$R_1 = \{A, B\}$

$R_2 = \{C, D\}$

$R_3 = \{A, C\}$ ← added to make decomposition lossless

$R = \{\underline{Sno}, Sname, City, \underline{Ino}, Iname, Prize\}$

$F = \{Sno, Ino \to Supplied\_Items, Sno \to Sname, Sno \to City, Ino \to Iname\}$

$F' = \{Sno, Ino \to Sname, \ldots, Sno, City \to Price, Sno \to Sname, Sno \to City, Ino \to Iname\}$

$F' = \{Sno, Ino \to Price, Sno \to Sname, City, Ino \to Iname\}$ → gives 3 3NF tables

Summary

- FDs give clues toward elimination of some redundancies in a rel^n schema
- goals: decompose rel^n schemas s.t. the decomposition is:
  1. lossless-join
  2. dependency-preserving
  3. BCNF (and if we fail here, at least 3NF)