3rd normal form (3NF)

- Non-prime attr. depend *only* on candidate keys
  - Consider FD X -> A
  - Either X is a superkey OR A is *prime* (part of a key)

- Counter-example:
  - studio -> studioAddr

    (*studioAddr* depends on *studio* which is not a candidate key)

Title	Year	Studio	StudioAddr
Star Wars	1977	Lucasfilm	1 Lucas Way
Patriot Games	1992	Paramount	Cloud 9
Last Crusade	1989	Lucasfilm	1 Lucas Way

Boyce-Codd normal form (BCNF)

- One additional restriction over 3NF
  - All non-trivial FDs have superkey LHS
- Counterexample
  - CanadianAddress(<u>street</u>, <u>city</u>, <u>province</u>, postalCode)
  - Candidate keys: {street, postalCode}, {street, city, province}
  - FD: postalCode -> city, province
  - Satisfies 3NF: city, province both prime
  - Violates BCNF: postalCode is not a superkey

=> Possible anomalies involving postalCode

Boyce-Codd Normal Form

- We say a relation *R* is in *BCNF* if whenever $X \rightarrow A$ is a nontrivial FD that holds in *R*, *X* is a superkey.
  - Remember: *non-trivial* means A is not contained in *X*.

Example: a relation not in BCNF

Drinkers(<u>name</u>, addr, <u>beersLiked</u>, manf, favBeer)

FD's: name→addr, favBeer,   beersLiked→manf

- Only key is {name, beersLiked}.
- In each FD, the left side is **not** a superkey.
- Any one of these FDs shows *Drinkers* is not in BCNF

Another Example

Beers(name, manf, manfAddr)

FD's: name->manf, manf->manfAddr

- Beers w.r.t. name->manf does not violate BCNF, but manf->manfAddr does.

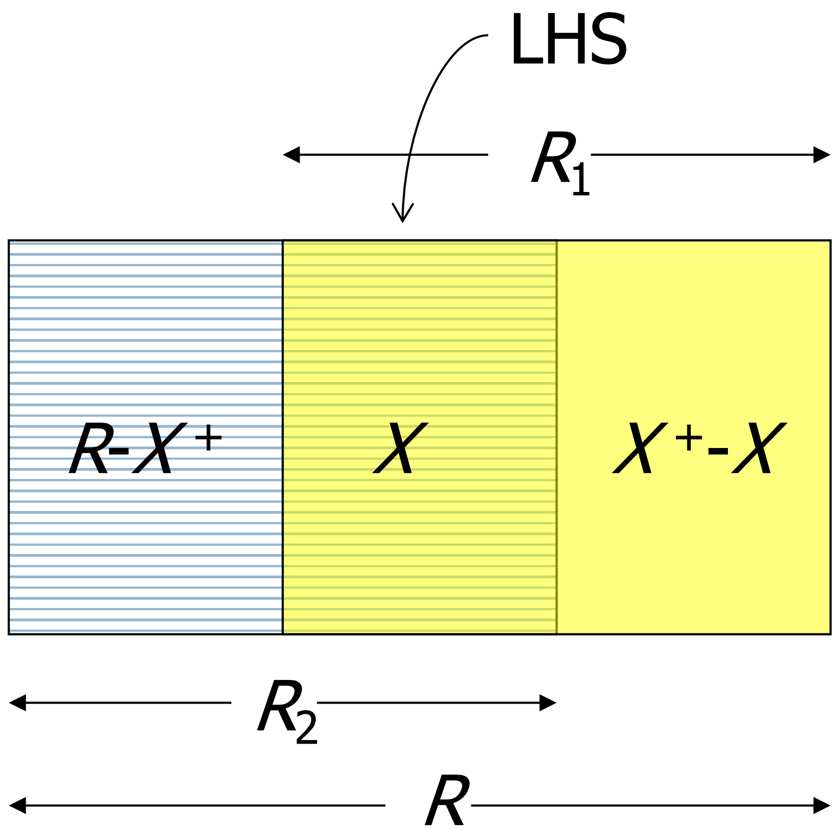
In other words, BCNF requires that:
the only FDs that hold are the result of key(s).
Why does that help?

Decomposition into BCNF

- Given: relation R with FDs F
- Look among the given FDs for a BCNF violation $X \rightarrow Y$ (i.e., X is not a superkey)

- Compute X^+.
  - Find $X^+ \neq X \neq$ all attributes, (o.w. X is a superkey)
- Replace R by relations with:
  - $R_1 = X^+$.
  - $R_2 = R - (X^+ - X) = R - X^+ \cup X$
- Continue to recursively decompose the two new relations
- *Project* given FDs F onto the two new relations.

Decomposition on X → Y



Example: BCNF Decomposition

Drinkers(<u>name</u>, addr, <u>beersLiked</u>, manf, favBeer)

F = {name→addr, name → favBeer, beersLiked→manf}

Key = name, beersLiked

- Pick BCNF violation name→addr.
- Closure: $\{name\}^+$ = {name, addr, favBeer}.
- Decomposed relations:
  - Drinkers1(<u>name</u>, addr, favBeer)
  - Drinkers2(<u>name</u>, <u>beersLiked</u>, manf)

Example -- Continued

- We are not done; we need to check Drinkers1 and Drinkers2 for BCNF.
- Projecting FDs is easy here.
- For Drinkers1(name, addr, favBeer), relevant FDs are name→addr and name→favBeer.
  - Thus, {name} is the only key and Drinkers1 is in BCNF.

Example -- Continued

- For Drinkers2(<u>name</u>, <u>beersLiked</u>, manf), the only FD is beersLiked→manf, and the only key is {name, beersLiked}.
  - Violation of BCNF.
- beersLiked$^+$ = {beersLiked, manf}, so we decompose *Drinkers2* into:
  - Drinkers3(<u>beersLiked</u>, manf)
  - Drinkers4(<u>name</u>, <u>beersLiked</u>)

Example -- Concluded

- The resulting decomposition of *Drinkers* :
  - Drinkers1(<u>name</u>, addr, favBeer)
  - Drinkers3(<u>beersLiked</u>, manf)
  - Drinkers4(<u>name</u>, <u>beersLiked</u>)
- Notice: *Drinkers1* tells us about drinkers, *Drinkers3* tells us about beers, and *Drinkers4* tells us the relationship between drinkers and the beers they like.

What we want from a decomposition

- *Lossless Join* : it should be possible to project the original relations onto the decomposed schema, and then reconstruct the original,
  i.e., get back exactly the original tuples.
- *No anomalies*
- *Dependency Preservation* : All the original FDs should be satisfied.

What we get from a BCNF decomposition

- *Lossless Join* : ✓
- *No anomalies* : ✓
- *Dependency Preservation* : ✗

Example: Failure to preserve dependencies

- Suppose we start with R(A,B,C) and FDs
  - $AB \rightarrow C$ and $C \rightarrow B$.
- There are two keys, {A,B } and {A,C }.
- $C \rightarrow B$ is a BCNF violation, so we must decompose into AC, BC.

The problem is that if we use AC and BC as our database schema, we cannot enforce the FD $AB \rightarrow C$ in these decomposed relations.