**Armstrong's Axioms**  
In these rules, assume W, X, Y, and Z are sets of attributes from a relation.

**Basic Rules**

1. Reflexivity: If , then X 🡪 Y.
2. Augmentation: If X 🡪 Y, then XW 🡪 YW.
3. Transitivity: If X 🡪 Y and Y 🡪 Z, then X 🡪 Z.

**Other rules** (technically all of these can be derived from the basic rules)

1. Splitting rule: If X 🡪 YZ, then X 🡪 Y and X 🡪 Z.
2. Combining rule: If X 🡪 Y and X 🡪 Z, then X 🡪 YZ.
3. Augmentation on the left: If X 🡪 Y, then XW 🡪 Y.

**Algorithm for closure of a set of attributes**

* Suppose you have a set of attributes {A1, …, An} and a set of FDs S.
* The closure of {A1, …, An} under S is the set of attributes B such that
  + every relation in S also satisfies A1…An -> B.
* Intuitive def'n: B is the largest set of attributes that we can deduce from knowing A1, …, An.
* Closure of {A1,…An} denoted by {A1,…An}**+**
* Hand-wavy algorithm (best kind!) ☺
  + Start with the set of attributes you're taking the closure of. Call that set X.
  + Look for a new FD where all the things on the left side on the FD are in X, but there's at least one attribute on the right that's not in X.
  + Add all the attributes on the right into X.
  + Repeat until you can't do this anymore (you can't find another FD to make it work).

**Algorithm for closure of a set of FDs**

* Repeatedly apply Armstrong's axioms until you can't find any more FDs.
* Hint: Start by splitting everything so all FDs have one attribute on the left only.
* Use transitivity and augmentation a lot.

**Algorithm for projecting a set of FDs**

* Given a set of FDs F, a starting relation R, and a subset of attributes from R, find all the FDs that hold using only the subset of attributes. Here we call the subset of attributes a new relation S.
* Compute closure F+. The projection is the set of all FDs in F+ that only involve attributes in S.

**BCNF**

* Anomalies are guaranteed not to exist when a relation is in ***Boyce-Codd normal form*** (BCNF).
* A relation R is in BCNF iff whenever there is a nontrivial FD A1…An->B1…Bm for R, {A1, …, An} is a superkey for R.
* Informally, the left side of every nontrivial FD must be a superkey.

**Checking for BCNF violations**

* List all nontrivial FDs in R.
* Ensure left side of each nontrivial FD is a superkey.
* (First have to find all the keys!)  
    
  Note: a relation with two attributes is always in BCNF.

**BCNF Decomposition**

Algorithm: Given relation R and set of FDs F:

* Check if R is in BCNF, if not, do:
* If there are FDs that violate BCNF, call one   
  X -> Y. Compute X+. Let R1 = X+ and R2 = X and all other attributes not in X+.
* Compute FDs for R1 and R2 (projection algorithm for FDs).
* Check if R1 and R2 are in BCNF, and repeat if needed.

**3NF**

* A relation R is in 3NF iff for every nontrivial FD A1…An -> B for R, one of the following is true:
  + A1…An is a superkey for R (BCNF test)
  + Each B is a ***prime*** attribute (an attribute in *some* key for R)

**3NF Decomposition**

* Given a relation R and set F of functional dependencies:

1. Find a minimal basis, G, for F.
2. For each FD X -> A in G, use XA as the schema of one of the relations in the decomposition.
3. If none of the sets of schemas from Step 2 is a superkey for R, add another relation whose schema is a key for R.