Functional Dependencies:

The evils of redundancy:

* Some information is stored repeatedly in the database
* The ROOT of several problems associated with relational schemas

What problems arise because of redundancy:

* ANOMALIES:
  + UPDATE anomalies
  + INSERTION anomalies
  + DELETION anomalies

Update:

* In Person(SSN, Name, Address, Hobby), if someone moves, only one tuple is updated
* New hobby added for someone with new address
  + Data inconsistency

Delete:

* Delete someone’s hobby by deleting their tuple
  + Address info no longer exists in database

FUNCTIONAL DEPENDENCY:

* Databases allow you to say that one attribute determines another
* X -> Y holds over relation schema R if, for every instance r of R:
  + For any two tuples t1, t2 in r,
  + 𝛑X (t1) = 𝛑X (t2) implies 𝛑Y (t1) = 𝛑Y (t2)
  + Where t1 and t2 are tuples; X and Y are sets of attributes
* In other words: X-> Y means
  + Given any two tuples in r, if the X values are the same, then the Y values must also be the same. (but not vise versa)
  + Can read -> as determines

FD’s Continued:

* An FD is a statement about all allowable relations
  + Must be identified based on semantics of application
* If K -> all attributes of R” then K is a superkey for R
* FDs are a generalization of keys

RULES OF INFERENCE:

* Armstrongs Axioms (AA): (X, Y, Z are sets of attributes)
  + Reflexivity: If X⊇Y the X-> Y
  + Augmentation: if X -> Y then XZ-> YZ for any Z
  + Transitivity: if X->Y and Y->Z, then X->Z
* Some additional rules:
  + Union: if X->Y and X->Z then X->YZ
  + Decomposition: if X->YZ, then X->Y and X->Z

Closure of FDs:

* An FD f is implied by a set of FDs F if f holds whenever all FDs in F hold
* F+ = closure of F is the set of all FDs that are implied by F(includes trivial dependencies)

Computing FD Closure:

* Typically we want to check if a given FD X->Y can be implied from a given set of FDs F
* It is equivalent to checking whether X->Y is in F+
* An efficient check:
  + Compute attribute closure of X (denoted X+ ) wrt F
  + X+ = set of attributes A such that X-> A is in F+
    - Initialize X+ := x
    - Repeat until no change: if there is an FD U->V in F such that U is in X+, then add V to X+
  + Check if Y is in X+ (i.e X->Y is in F+)

FD Closure vs Finding Key:

* Computing FD Closure can be used to find the keys of a relation
  + If X+ = {all attributes of R}, then X is a superkey for R.
  + Question: How to check if X is a candidate key?
  + Answer: check whether any subset Y of X satisfies: Y+ = {all attributes of R}

How to determine Candidate Keys?

* An efficient solution:
  + When computing FD closure, we distinguish attributes into three categories:
    - L: attributes only appear at the left side of all given FDs
    - R: attributes only appear at the right side of all given FDs
    - M: attributes that appear on both sides of all given FDs
  + The Principle:
    - Attributes in L: each candidate key should include all attributes in L
    - Attributes in M may be part of keys
    - Attributes in R will NOT be part of any key