A few details … using Armstrong’s axioms

Supplement to Normalization Lecture

Lois Delcambre

Armstrong's Axioms – with explanation and examples

Reflexivity: If X $\supseteq$ Y, then X$\rightarrow$Y. (identity function is a function)

Augmentation: If X$\rightarrow$Y, then XZ$\rightarrow$YZ, for any Z. (parallel application of one function and the identity function is a function)

Transitivity: If X$\rightarrow$Y and Y$\rightarrow$Z, then X$\rightarrow$Z. (composition of two functions is a function)

Examples:

Reflexivity: ssn$\rightarrow$ssn, ssn,name$\rightarrow$ssn

Augmentation: If ssn$\rightarrow$name then ssn,color$\rightarrow$name,color

Transitivity: If ssn$\rightarrow$mgr-id and mgr-id$\rightarrow$mgr-name, then ssn$\rightarrow$mgr-name.

Using Armstrong's Axioms

Reflexivity: If $X \supseteq Y$, then $X \to Y$.

Augmentation: If $X \to Y$, then $XZ \to YZ$, for any Z.

Transitivity: If $X \to Y$ and $Y \to Z$, then $X \to Z$.

Decomposition: If $X \to YZ$, then $X \to Y$ and $X \to Z$

Proof:

$X \to YZ$	given
$YZ \to Y$	Reflexivity (trivial FD)
$X \to Y$	Transitivity

(Similarly, $X \to Z$.)

Using Armstrong's Axioms (cont.)

Proposition - Superkeys can be derived from keys:

If $X \rightarrow Y$, then $XA \rightarrow Y$, for any A

Example: If SSN $\rightarrow$ name, then SSN,color $\rightarrow$ name

Proof:

$X \rightarrow Y$	Given
$XA \rightarrow YA$	Augmentation
$XA \rightarrow Y$	Decomposition (proved on previous slide)

I can construct a superkey from a key.

Using Armstrong's Axioms (cont.)

Proposition: If $X \rightarrow A$ and X is a superkey (and not a key) for the table, then $X \rightarrow A$ is derivable from a key.

Proof:

$X \rightarrow A$	Given
$X = Y \cup Z$ where:	
Y is a key,	
Z is non-empty,	
Y and Z disjoint	Because X is a superkey but not a key
$Y \rightarrow A$	Because Y is a key for the table that A is in

Therefore $X \rightarrow A$ follows from: $Y \rightarrow A$ (implied by the key) and the result from the previous slide.

Formal definition of BCNF
(in the textbook) - revisited

- For a table R, every FD X $\longrightarrow$ A that occurs among attributes of R then either:
  - A is an element of X (X $\longrightarrow$ A is trivial)
  - ~~A is part of a key (don't worry about "key" attributes)~~
  - X is a superkey of R
    consider the following 2 cases:
    - X is a key for R (good)
    - X is a superkey for R (and not a key). The X $\longrightarrow$ A is derivable from a key using augmentation and decomposition.

For a table to be in BCNF, every FD is either trivial or derivable from the FDs implied by the key(s).

Informally, I often say, BCNF if all FDs are implied by the key(s).

Formal definition of 3NF
(in the textbook)

- For a table R, every FD $X \to A$ that occurs among attributes of R then either:
  - A is an element of X ($X \to A$ is trivial)
  - A is part of a key (ignore the "key" attributes)
  - X is a superkey of R
    Consider the following 2 cases:
    - X is a key for R (good)
    - X is a superkey for R (and not a key). The $X \to A$ is derivable from a key using augmentation. (Stay tuned.)

A table is in 3NF if all the non-key attributes are either trivial or implied by FDs derivable from the FDs implied by the key(s).

Using Armstong's Axioms to show dependency preservation (that SSN $\rightarrow$dname is not lost)

Employee (<u>SSN</u>, name, phone, dept, dept-name)

Employee (<u>SSN</u>, name, phone, dept)

Department (<u>dept</u>, dname)

F = {SSN$\rightarrow$name, SSN$\rightarrow$phone, SSN$\rightarrow$dept, SSN$\rightarrow$dept-name, dept$\rightarrow$dname} original set of FDs

G = {SSN$\rightarrow$name, SSN$\rightarrow$phone, SSN$\rightarrow$dept, ~~SSN $\rightarrow$dept-name~~, dept$\rightarrow$dname} the set of FDs projected from F

But G^+ includes SSN$\rightarrow$dept-name because we can derive it:

SSN$\rightarrow$dept	Given (it is in G)
dept$\rightarrow$dname	Given (it is in G)
SSN$\rightarrow$dname	Because of transitivity.

Example showing that we must project from F^+ when considering dependency preservation

R(<u>a, b</u>, c) where ab is a key, a $\rightarrow$b, b$\rightarrow$a, a $\rightarrow$c

F = {ab$\rightarrow$c, b$\rightarrow$a, a$\rightarrow$b, a$\rightarrow$c}

Suppose we decompose to
X(a, b) and Y(b, c)

If we project F onto X and Y, we see:
a$\rightarrow$b, b$\rightarrow$a and that's it. We appear to have lost many FDs.

But F^+ includes this additional FD:
b$\rightarrow$c (because b$\rightarrow$a and a$\rightarrow$c)

If we project F^+ onto X and Y we see:
a$\rightarrow$b, b$\rightarrow$a , and b$\rightarrow$c in G.

G^+ then includes a$\rightarrow$b, b$\rightarrow$a , b$\rightarrow$c, plus a$\rightarrow$c (by transitivity), ab$\rightarrow$c (by augmentation).

Same example using realistic attribute names

R(<u>ssn, id</u>, name)

where (ssn, id) is a key,

F = {ssn $\rightarrow$id, id$\rightarrow$ssn, ssn$\rightarrow$name}

X(<u>ssn</u>, <u>id</u>)   Y(<u>id</u>, name)

Projection of F = {ssn $\rightarrow$id, id$\rightarrow$ssn}

But F^+ includes id$\rightarrow$name (because id $\rightarrow$ssn, and ssn $\rightarrow$name)

Projection of F^+ includes {ssn $\rightarrow$id, id$\rightarrow$ssn, id $\rightarrow$name}

From that projection, we can compute G+ which includes
ssn$\rightarrow$name (because ssn$\rightarrow$id, id$\rightarrow$name)

Challenge Question

Proposition:

If $AB \rightarrow C$ (where A and B are disjoint sets of attributes)

then $A \rightarrow C$ and $B \rightarrow C$.

Is this true or false?

Can you prove/disprove it?

Other Normalization Results

- When a table is in BCNF, it is not possible to have redundancies or update anomalies caused by FDs.
- There are other dependencies besides FDs.
  - Multi-valued dependency … leads to the definition of 4NF
  - Join dependency … leads to the definition of 5NF

- For FDs, MVDs, and JDs, the project operator is used to decompose and the join operator is used to recontruct.
- There are no redundancies or update anomalies that remain in a table in 5NF that can be solved by projecting/joining. 5NF is sometimes PJNF.

Normalization made easy

Every attribute in a table must depend on the
key (definition of a key),
the whole key (2NF – no partial dependencies)
and
nothing but the key
(3NF – no transitive dependencies).

Every non-key attribute in a table must depend on the
key (definition of a key)
the whole key (2NF – no partial dependencies)
and
nothing but the key
(3NF – no transitive dependencies).