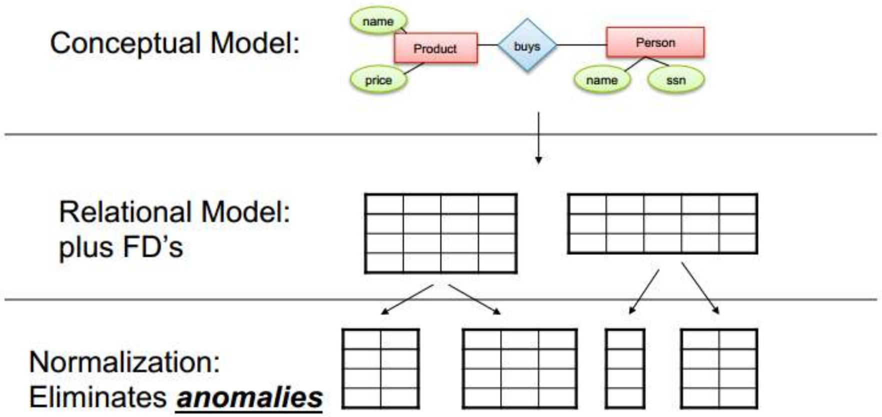
Introduction to Data Management CSE 344

Section 8 – Boyce Codd Normal Form

Part I --- Conceptual Design

Normal forms and functional dependencies:

- Anomalies(redundancy, update/deletion anomalies), functional dependencies, attribute closures, BCNF decomposition



- The BCNF (Boyce-Codd Normal Form) ---- A relation R is in BCNF if every set of attributes is either a superkey or its closure is the same set.

Example 1.

Consider the following relational schema and set of functional dependencies. R(A,B,C,D,E,F,G) with functional dependencies:

A -->D

D --> C

F --> EG

DC --> BF

Decompose R into BCNF.

Example 1 -- Solution.

R(<u>A</u>,B,C,D,E,F,G)

A -->D
D --> C
F --> EG
DC --> BF

Solution: Watch-out! The first FD does NOT violate BCNF so we need to pick another one to decompose. We try the second one:
Try $\{D\}^+ = \{B, C, D, E, F, G\}$. Decompose into R1(B, C, <u>D</u>, E, F, G) and R2(<u>A</u>,D).

R2 has two attributes, so it is necessarily in BCNF.

For R1, again not all FDs violate BCNF so we need to be careful.
Try $\{F\}^+ = \{E, F, G\}$. Decompose into R11(E, <u>F</u>, G) and R12(B, C, <u>D</u>, F).

Both R11 and R12 are in BCNF.

Example 2.

Relation $R(A,B,C,D,E,F)$ and functional dependencies:

$A \rightarrow BC$ and $D \rightarrow AF$

Decompose R into BCNF.

Example 2 -- Solution.

Relation R(A,B,C,D,E,F) and FD's A → BC and D → AF

A→BC violates BCNF since A+ = ABC ≠ ABCDEF. So we split R into R1(ABC) and R2(ADEF).

The only non-trivial FD in R1 is A→BC, and A+ = ABC, so R1 is in BCNF.

R2 has a non-trivial dependency D→AF that violates BCNF because D+ = ADF ≠ ADEF. So we split R2 into R21(DAF) and R22(DE). Both of these are in BCNF since they have no non-trivial dependencies that are not superkeys.

Example 3

Relational schema: R(A,B,C,D,E),

functional dependencies: AB—>C, BC—>D

Decompose R into BCNF.

Example 3 -- solution

Relational schema: R(A,B,C,D,E),

functional dependencies: AB—>C, BC—>D

First step uses BC+=BCD and decomposes into R1(B,C,D), R2(A,B,C,E); second step decomposes R2 into R3(A,B,C) and R4(A,B,E)

Example 4

The relation is R (A, B, C, D, E) and the FDs :

A -> E, BC -> A, and DE -> B

Decompose R into BCNF.

Example 4 – solution 1

The relation is R (A, B, C, D, E) and the FDs :

A -> E, BC -> A, and DE -> B

Notice that {A}+ = {A,E}, violating the BCNF condition. We split R to R_1(A,E) and R_2(A,B,C,D).

R_1 satisfies BCNF now, but R_2 not because of: {B,C}+ = {B,C,A}. Notice that the fd D E -> B has now disappeared and we don't need to consider it! Split R_2 to: R_2A(B,C,A) and R_2B(B,C,D).

Example 4 – solution 2

The relation is R (A, B, C, D, E) and the FDs :

A -> E, BC -> A, and DE -> B

Can we split differently? Let's try with the violation {B,C}+ = {B,C,A,E}. We initially split to R_1(B,C,A,E) and R_2(B,C,D). Now we need to resolve for R_1 the violation {A}+ = {A,E}. So we split again R_1 to R_1A(A,E) and R_1B(A,B,C). The same!

We can also start splitting by considering the BCNF violation {D,E}+ = {D,E,B}. Which is the resulting BCNF decomposition in this case? (it will be a different one)

Part II -- Lossless-join decomposition

Consider the relation R(A,B,C,D,E)

with FDs: {AB -> C, BC -> D, AD -> E}. We want to check whether the decomposition {ABC, BCD, ADE} is a lossless-join decomposition.

Part II -- Lossless-join decomposition

Consider the relation R(A,B,C,D,E)
with FDs: {AB -> C, BC -> D, AD -> E}. We want to check whether the decomposition {ABC, BCD, ADE} is a lossless-join decomposition.

Start by constructing a tableau as follows:

A	B	C	D	E
a	b	c	d1	e1
a1	b	c	d	e2
a	b1	c1	d	e

Part II -- Lossless-join decomposition

```
A  | B  | C  | D  | E
------------------------------
a  | b  | c  | d1 | e1
a1 | b  | c  | d  | e2
a  | b1 | c1 | d  | e
```

BC -> D

```
A  | B  | C  | D  | E
------------------------------
a  | b  | c  | d  | e1
a1 | b  | c  | d  | e2
a  | b1 | c1 | d  | e
```

Notice that we use a common distinguished variable (a,b,c,...) if the variable is a key, otherwise we use a non-distinguished symbol (e1, e2, b1,...) We next start applying the fd's! Notice that the 1st and 2nd row have the same distinguished B and C attributes. Hence, D must be the same by the fd BC -> D. This results in unifying d1 = d. Now the table becomes:

Part II -- Lossless-join decomposition

But now rows 1 and 3 agree on A and D. Because AD -> E, we unify e1 = e. Now, we have:

A	B	C	D	E
a	b	c	d	<u>e1</u>
a1	b	c	d	e2
a	b1	c1	d	e

AD -> E

A	B	C	D	E
a	b	c	d	e
a1	b	c	d	e2
a	b1	c1	d	e

Part II -- Lossless-join decomposition

A | B | C | D | E

----

a | b | c | d | e
a1 | b | c | d | e2
a | b1 | c1 | d | e

Row 1 contains only distinguished symbols, hence the algorithm terminates and the answer is YES, the decomposition is lossless. If we could not apply any fd and no row had only distinguished symbols, we would terminate with NO. This method is called the "chase".