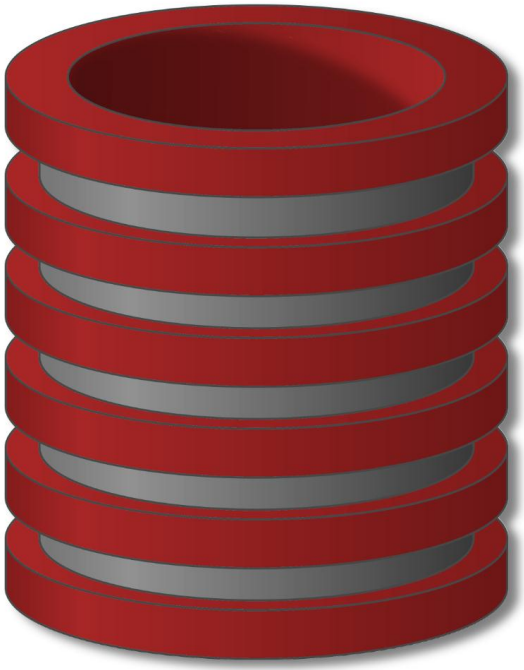


Relational Design Theory

Boyce-Codd Normal Form

Jennifer Widom

Relational design by decomposition

- "Mega" relations + properties of the data
- System decomposes based on properties
- Final set of relations satisfies normal form
  - No anomalies, no lost information
- Functional dependencies ⇒ Boyce-Codd Normal Form
- Multivalued dependences ⇒ Fourth Normal Form

Decomposition of a relational schema

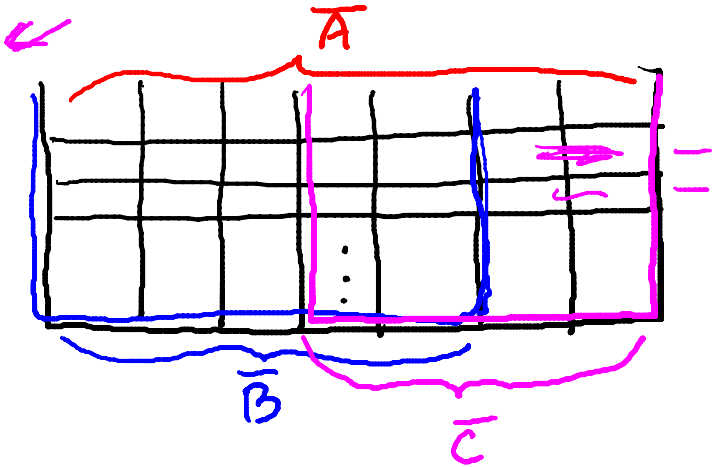
$R(A_1, \ldots, A_n)$ $\bar{A}$

$R_1(B_1, \ldots, B_k)$ $\bar{B}$ $\quad \bar{B} \cup \bar{C} = \bar{A}$ ✓

$R_2(C_1, \ldots, C_m)$ $\bar{C}$ $\quad R_1 \bowtie R_2 = R$ ✓

$R_1 = \Pi_{\bar{B}}(R)$

$R_2 = \Pi_{\bar{C}}(R)$



Decomposition Example #1

Student(SSN, sName, address,
HScode, HSname, HScity, GPA, priority)

S1(SSN, sName, addr, HScode, GPA, priority)

S2(HScode, HSname, HScity)

$\bar{A} \cup \bar{B} = \bar{C}$ S1 ⋈ S2 = Student

Decomposition Example #2

```
Student(SSN, sName, address,
        HScode, HSname, HScity, GPA, priority)
```

S1 (SSN, sName, addr, HScode, HSname, HScity)

S2 (sName, HSname, GPA, priority)

$\overline{A} \cup \overline{B} = \overline{C}$

S1 ⋈ S2 = Student ?

Relational design by decomposition

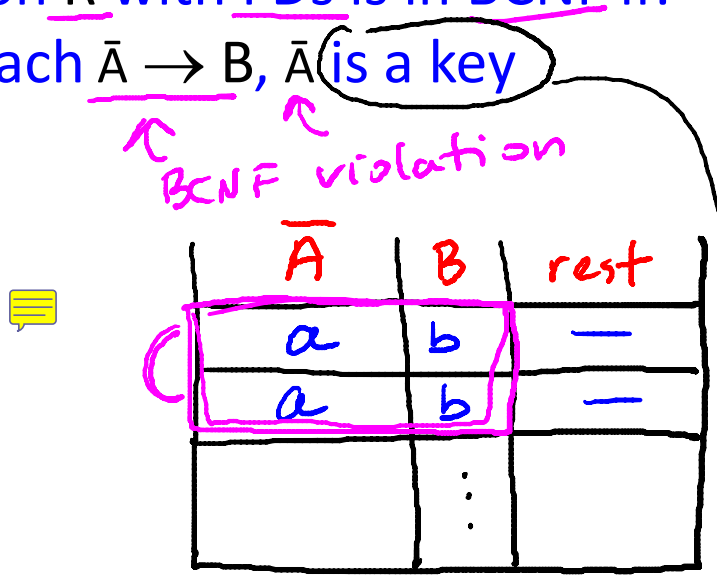
- "Mega" relations + properties of the data
- System decomposes based on properties

- "Good" decompositions only ← "reassembly" produces orig. Lossless join property
- Into "good" relations ← BCNF

Boyce-Codd Normal Form

Relation R with FDs is in BCNF if:

For each $\bar{A} \rightarrow B$, $\bar{A}$ is a key



$\bar{A} \rightarrow B$

$\bar{A}$ ⊂ key

$\bar{A}$ C D

"contains a key"

"is a superkey"

BCNF? Example #1

Student(SSN, sName, address,
HScode, HSname, HScity, GPA, priority)

- SSN → sName, address, GPA
- GPA → priority
- HScode → HSname, HScity

Keys: {SSN, HScode}

Every FD have a key on LHS?

No! No

BCNF? Example #2

Apply(SSN, cName, state, date, major)

SSN, cName, state → date, major

key

In BCNF.

Relational design by decomposition

- "Mega" relations + properties of the data
- System decomposes based on properties

❖ "Good" decompositions only — algorithm.

❖ Into "good" relations ✓ BCNF

BCNF decomposition algorithm

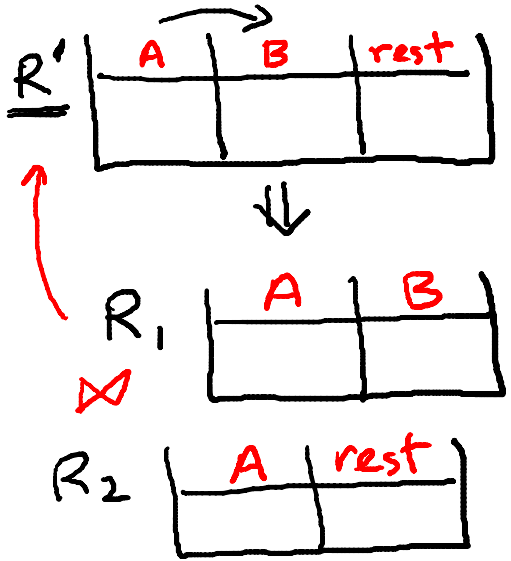
Input: relation R + FDs for R

Output: decomposition of R into BCNF relations with "lossless join"

Compute keys for R using FDs

Repeat until all relations are in BCNF:

- Pick any R' with $\overline{A} \rightarrow \overline{B}$ that violates BCNF
- Decompose R' into $R_1(A, B)$ and $R_2(A, rest)$
- Compute FDs for R_1 and R_2
- Compute keys for R_1 and R_2



BCNF Decomposition Example

Student(SSN, sName, address, HScode, HSname, HScity, GPA, priority)

SSN → sName, address, GPA

GPA → priority

HScode → HSname, HScity

Key: {SSN, HScode}

S1(HScode, HSname, HScity)

~~S2(SSN, sName, addr, HScode, GPA, priority)~~

S3(GPA, priority)

~~S4(SSN, sName, addr, HScode, GPA)~~

S5(SSN, sName, addr, GPA)

S6(SSN, HScode)

BCNF decomposition algorithm

Input: relation R + FDs for R

Output: decomposition of R into BCNF relations with "lossless join"

Compute keys for R

Repeat until all relations are in BCNF:

- Pick any R' with $A \rightarrow B$ that violates BCNF
- Decompose R' into $R_1(A, B)$ and $R_2(A, rest)$
- Compute FDs for R_1 and R_2
- Compute keys for R_1 and R_2

different answer

"Extend"

$A \rightarrow B$

$A \rightarrow BA^+$

Implied FDs Closure.

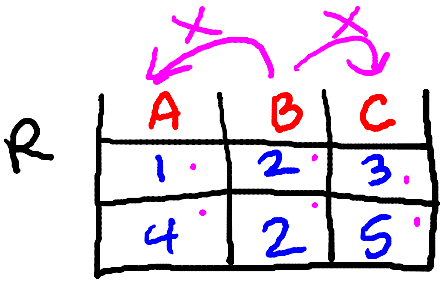
Does BCNF guarantee a good decomposition?

- Removes anomalies? ✓
- Can logically reconstruct original relation?

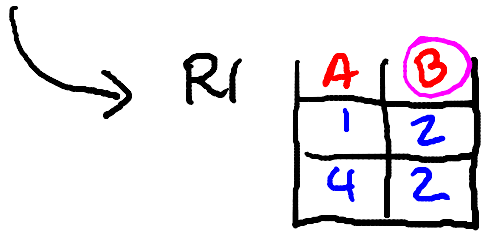
  Too few or too many tuples?

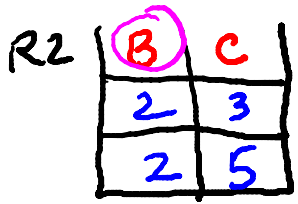
yes

R → R1 ⋈ R2



R1 ⋈ R2 =
123
125
423
425





Does BCNF guarantee a good decomposition?

- Removes anomalies?
- Can logically reconstruct original relation?

  Too few or too many tuples?
- Some shortcomings discussed in later video