$R = \{A_1, \ldots, A_n\} \cup \{B_1, \ldots, B_n\}$

$F = \bigcup_{1 \le i \le n} \{A_i \to B_i, B_i \to A_i\}$

$X = \left\{ \begin{bmatrix} A_1 \\ B_1 \end{bmatrix}, \begin{bmatrix} A_2 \\ B_2 \end{bmatrix}, \ldots, \begin{bmatrix} A_n \\ B_n \end{bmatrix} \right\}$

↳ exponential # of candidate keys in $\lg(n)$

$\{\{A_1, \ldots, A_n\} \to \{B_1, \ldots, B_n\}\}$

$\{\{A_1, \ldots, A_n\} \to \{B_1\},$
$\quad \ldots \quad \to \{B_n\}\}$

Conservative extension: Introduction of extra columns without introducing feedback (?)

$R = \{A_1, \ldots, A_n, B_1, \ldots, B_n\} \cup \{C\}$

$\{\{A_1, \ldots, A_n\} \to \{C\},$
$\{C\} \to \{B_1\},$
$\ldots$
$\{C\} \to \{B_n\}\}$

→ quadratic blowup

→ Allows more restrictive grammar for

functional dependencies.

func Compute X^+ (X, F):

$X^+ := X$

while true:

if $\exists (Y \to Z) \in F$ such that

(1) $Y \subseteq X^+$, and

(2) $Z \notin X^+$

then $X^+ := X^+ \cup Z$

else exit

return X^+

Loop invariant: $X \to X^+ \in F$

or $\forall A \in X^+ \cdot X \to A \in F^+$

Definition (Decomposition):

Let R be a relational schema (attributes).

The collection $\{R_1, \ldots, R_n\}$ of schemas is a **decomposition of R** if

$$R = R_1 \cup R_2 \cup \cdots \cup R_n$$

A good decomposition does **not**:

- lose information
- complicate checking of constraints → efficient to check FDs satisfied by materialized views
- contain anomalies (or at least contains fewer anomalies than R)

1: We should be able to construct the instance of the original table from the instances of the tables in the decomposition

Lossless-Join Decomposition

Natural join produces spurious tuples and thus loses information (example in slides)

Definition

A decomp. $\{R_1, R_2\}$ is <u>lossless</u> iff the common attributes of R_1 and R_2 form a superkey for either schema, that is

$$R_1 \cap R_2 \to R_1 \quad \text{or} \quad R_1 \cap R_2 \to R_2$$

e.g. $R = \{Student, Asn, Group, Mark\}$

$F = \{(Student, Asn \to Group, Mark)\}$

$R_1 = \{Student, Grp, Mark\}$

$R_2 = \{Asn, Mark\}$

$\Rightarrow$ lossy decomposition b/c

$R_1 \cap R_2$ $(= \{M\})$ is not a superkey of either SGM or AM

To check this:

either $R_1 \subseteq$ Compute $X^+(R_1 \cap R_2, F)$

or $R_2 \subseteq$ Compute $X^+(R_1 \cap R_2, F)$

$(R_1 \cap R_2 \xrightarrow{\text{by theorem}} R_1 \in F^+$ or vice versa$)$

Dependency Preservation (2nd criterion)

How do we test/enforce constraints on the decomposed schema?

Definition

A decomposition $D = \{R_1, \ldots, R_n\}$ is dependency preserving if there is an equivalent set of F' of functional dependencies, none of which are interrelational with D.

Is $F'^+ = F^+$?

$\equiv$ (a) $\forall X \to Y \in F'$ is $X \to Y \in F^+ \Rightarrow F'^+ \subseteq F^+$

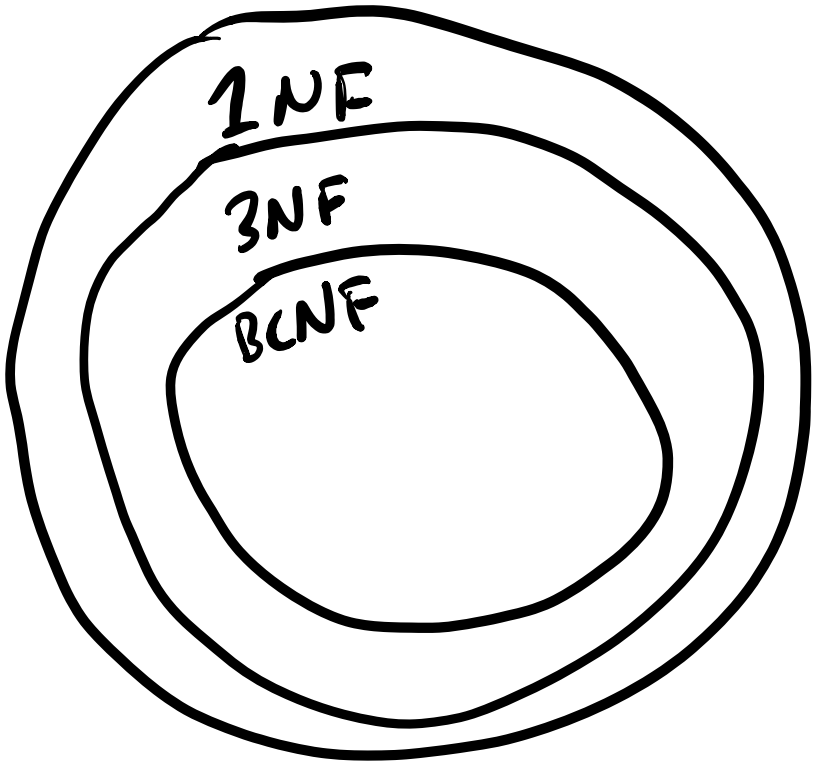
(b) $\forall X \to Y \in F$ is $X \to Y \in F'^+ \Rightarrow F^+ \subseteq F'^+$

Avoiding Anomalies (3rd criterion)

roughly, each relation should consist of a primary key and a set of mutually independent tables

We discuss Boyce-Codd Normal Form (BCNF) and Third Normal Form (3NF).

Normal Forms



Hierarchical schema = Relational schema but table of nodes and foreign keys is a tree (in essence, JSON).

1NF: requires the value of every field to be a primitive from the domain

Definition (BCNF)

Schema R is in **BCNF w.r.t F** iff whenever $(X \to Y) \in F^+$ and $XY \subseteq R$ then either

- $(X \to Y)$ is trivial (i.e. $Y \subseteq X$), or
- X is a superkey of R.

A database schema $\{R_1, \ldots, R_n\}$ is in **BCNF** iff each R_i is in BCNF.

Formalizes the goal that independent relationships are stored in separate tables.

```
func ComputeBCNF(R, F):
    result := {R}
    while ∃Ri ∈ Result and (X → Y) ∈ F+
            violates the BCNF condition:
        replace Ri by Ri - (Y - X)
        add {X, Y} to result
    return result
```

- no efficient lossless-join BCNF decomp. algo. exists
- results depend on sequence of FDs used to decompose the R_is
- possible that no lossless join dependency

preserving BCNF decomp exists.

$R = \{A, B, C\}$ and $F = \{AB \to C, C \to B\}$

↳ can easily see interrelational dependency must exist (2 tables of 2 columns)

e.g.

Sno	Sname	City	Ino	Iname	Price

{ {Sno, Sname, City, Ino, Iname, Price} }

Not in BCNF.

Sno → Sname, City found

{Sno, Ino, Iname, Price}

In BCNF

{Sno, Sname, City}

Ino → Iname

{Sno, Ino, Price}

{Ino, Iname}

done.

3rd Normal Form

Schema R is in 3NF w.r.t. F iff $(X \to Y) \in F^+$ and $XY \subseteq R$ gives:

- $X \to Y$ trivial, or
- X is a superkey of R, or
- each attribute of Y contained in a candidate key of R.

A schema $\{R_1, \ldots, R_n\}$ is in 3NF iff each relation schema R_i is in 3NF.

- 3NF is looser than BCNF, allows more redundancy:

$\Rightarrow R = \{A, B, C\}$ and $F = \{AB \to C, C \to B\}$ in 3NF