# Dependency Preservation

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

In the last lecture, we considered the following scenario: decompose

$$R(A, B, C, D) \text{ under } F = \{A \rightarrow B, B \rightarrow C\}.$$

Our decomposition resulted in:

$$R_1(AB),\ R_2(AC), \text{ and } R_3(AD)$$

all of which are in BCNF.

These tables are very good when the database is static, namely, no tuple insertion will occur in the future. However, they have a defect when the database is dynamic:

**Think**

How do we check whether a tuple insertion violates:

- $A \rightarrow C$?
- $B \rightarrow C$?

Recall that no FD is allowed to be violated at any time.

## Dependency Preservation

A FD $X \rightarrow Y$ is preserved in a relation $R$ if $R$ contains all the attributes of $X$ and $Y$.

A FD can therefore be checked by accessing only $R$.

Example. In the previous slide:

- $A \rightarrow B$ is preserved in $R_1$.
- $B \rightarrow C$ is not preserved in any relation.

Let us revisit the scenario of decomposing

$$R(A, B, C, D) \text{ under } F = \{A \rightarrow B, B \rightarrow C\}.$$

Consider the following decomposed tables:

$$R_1(AB), \ R_2(BC), \text{ and } R_3(AD)$$

all of which are in BCNF.

This decomposition is better than the previous one because:

- Both $A \rightarrow B$ and $B \rightarrow C$ are preserved.
- Hence, each can be checked in one table (thus avoiding joins, which are typically slow).

**Note**

How about $A \rightarrow C$? It is not preserved, so how do we check it?

Let:

- $S$ be the set of tables in our final design.
- $F$ be the set of FDs we have collected from the underlying application.
- $F'$ be the set of FDs each of which is preserved in at least one table in $S$.

**Definition**

Our design $S$ is dependency preserving if $F'^{+} = F^{+}$.

In other words, by checking only the FDs in $F'$, we effectively have checked the entire $F^{+}$.

# Example 1

If we decompose

$$R(A, B, C, D) \text{ under } F = \{A \to B, B \to C\}.$$

into

$$R_1(AB),\ R_2(AC), \text{ and } R_3(AD),$$

then:

- $S = \{R_1, R_2, R_3\}$.
- $F' = \{A \to B,\ A \to C,$ (omitting trivial FDs)$\}$
- $F'^{+} \neq F^{+}$

Therefore, $S$ is not dependency preserving.

# Example 2

If we decompose

$$R(A, B, C, D) \text{ under } F = \{A \to B, B \to C\}.$$

into

$$R_1(AB),\ R_2(BC),\ \text{and}\ R_3(AD),$$

then:

- $S = \{R_1, R_2, R_3\}$.
- $F' = \{A \to B,\ B \to C,\ \text{(omitting trivial FDs)}\}$
- $F'^{+} = F^{+}$

Therefore, $S$ is dependency preserving.

When the database needs to be dynamic (i.e., tuple insertions may occur), we aim at achieving three principles:

1. Capture all the information that needs to be captured by the underlying application.
2. Achieve the above with little redundancy.
3. Make our design dependency preserving.

Unfortunately, it is not always possible to realize all principles simultaneously. See next.

Consider table SUPERVISE(profId, stuId, fypId) under the following FDs:

$$\text{stuId, fypId} \rightarrow \text{profId}$$
$$\text{profId} \rightarrow \text{fypId}$$

It is impossible to have a dependency preserving design with only BCNF tables because

- SUPERVISE is not in BCNF
- Any decomposition will fail to preserve "stuId, fypId $\rightarrow$ profId".