3NF

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

We have seen in the previous lecture that it is not always possible to have a BCNF design that is also dependency preserving. In other words, between BCNF and dependency preserving, we must choose one and compromise the other.

In practice, people usually hold onto dependency preserving, and give up BCNF.

Discuss

Why?

Let F be the set of functional dependencies (FD) we have collected, and F^+ the closure of F.

Definition

A relation R is in 3rd Normal Form (3NF) if F^+ has no FD $X \rightarrow A$ such that all the following hold:

1. A and X appear in R (i.e., the FD "concerns" R)
2. $A \notin X$ (i.e., the FD is not trivial)
3. X does not contain any candidate key of R
4. A does not belong to any candidate key of R.

A BCNF table is always in 3NF, but not the vice versa.

When it is not possible to have a BCNF design that is dependency preserving, people turn to make sure that all the tables are in 3NF. There are some very technical reasons (that go beyond the course), but the following is a good one: it is always possible to produce a 3NF design that is dependency preserving, as we will see. In fact, one can think of 3NF as a "minimal" relaxation of BCNF to always allow a dependency-preserving design.

Example. Consider table SUPERVISE(profId, stuId, fypId) under the set F of following FDs:

$$\begin{aligned} \text{stuId, fypId} &\rightarrow \text{profId} \\ \text{profId} &\rightarrow \text{fypId} \end{aligned}$$

Is the table in 3NF?

Answer. SUPERVISE has candidate keys {stuId, fypId} and {stuId, profId}. F^+ contains:

$$\begin{aligned} \text{profId} &\rightarrow \text{fypId} \\ \text{profId, stuId} &\rightarrow \text{fypId} \\ \text{stuId, fypid} &\rightarrow \text{profId} \end{aligned}$$

and other trivial FDs. It is now easy to verify that the table is in 3NF.

Example. Consider attributes A, B, C, D, and the set F of FDs:

$$A \to D$$
$$AB \to C$$
$$AD \to C$$
$$B \to C$$
$$D \to AB$$

Is $R(ABCD)$ in 3NF?

Answer. R has candidate keys A and D. Then, $B \to C$ determines that R is not in 3NF because:

- It concerns R.
- It is not trivial.
- Its left hand side does not contain any candidate key.
- Its right hand side is not contained in any candidate key.

We now proceed to explain how to carry out table decomposition in a dependency preserving manner. At a high level, there are two steps:

- Simplify the set F of FDs as much as possible
- Decompose using the simplified FDs.

Definition

A set F of FDs is all-regular if all FDs in F are regular (i.e., each FD has only one attribute on the right).

Example. The set of FDs in the example of Slide 6 is not all-regular. But it can be easily transformed into the following all-regular set:

$$A \to D$$
$$AB \to C$$
$$AD \to C$$
$$B \to C$$
$$D \to A$$
$$D \to B$$

Definition

A regular FD $X \to A$ is simpler than another regular FD $X' \to A$ if $X \subset X'$ (note that the two FDs need to have the same right hand side).

Example. $A \to C$ is simpler than $AB \to C$.

Definition

An all-regular set F of FDs is simplifiable if we can carry out one of the following without changing its closure:

- remove a FD from F
- make a FD of F simpler.

Example. Let F be the set of following FDs on A, B, C, D:

$$A \to C$$
$$B \to C$$
$$AB \to C$$

F is simplifiable because $AB \to C$ can be removed. More specifically, let F' be the set of the first two FDs; then $F'^{+} = F^{+}$.

Example. Let F be the set of following FDs on A, B, C, D:

$$A \to B$$
$$AB \to C$$

F is simplifiable because $AB \to C$ can be made simpler as $A \to C$. More specifically, let F' be the set of the FDs:

$$A \to B$$
$$A \to C$$

then $F'^{+} = F^{+}$.

Definition

An all-regular set F of FDs is minimal if it is not simplifiable.

Example. Let F be the set of following FDs on A, B, C, D:

$$A \to B$$
$$B \to C$$

Then, F is minimal.

Definition

Let F be an all-regular set of FDs. An all-regular set G of FDs is a minimal cover of F if

- $F^+ = G^+$
- G is minimal.

Minimal cover is sometimes also called canonical cover.

Example. Let F be the set of following FDs on A, B, C, D:

$$A \to B$$
$$AB \to C$$

Let G be the set of FDs:

$$A \to B$$
$$A \to C$$

G is a minimal cover of F.

3NF Decomposition

We are ready to elaborate on the algorithm for obtaining a 3NF design. Assume that we have already obtained a design that is a set S of BCNF tables (a BCNF design is always possible as long as we do not require it to be dependency preserving). As before, let F be the set of all-regular FDs we have collected from the underlying application.

3NF Decomposition

algo (S, F)
/* output: a set S' of 3NF tables which constitute a dependency-preserving design */

1. $G \leftarrow$ a minimal cover of F
2. $G' \leftarrow$ the set of FDs in G each of which is not preserved in any table of S
3. $S' \leftarrow S$
4. for each FD $X \rightarrow A$ in G'
5. add to S' a table with schema $X \cup \{A\}$
6. return S'

Example. Consider relation $R(A, B, C, D)$, and the set F of FDs:

$$A \to D$$
$$AB \to C$$
$$AD \to C$$
$$B \to C$$
$$D \to A$$
$$D \to B$$

Decompose R into 3NF tables.

Answer.

1. R has candidate keys A and D. By the BCNF decomposition algorithm we learned before, we can decompose R into BCNF tables $S = \{R_1(BC), R_2(ABD)\}$.

2. Compute a minimal cover G of F:

$$A \to D$$
$$A \to C$$
$$B \to C$$
$$D \to A$$
$$D \to B$$

3. Find G', i.e., the set of FDs not preserved in any table of S:

$$A \to C$$

4. Add table $R_3(AC)$ into our design.

R_1, R_2, R_3 constitute our final design. In this example, we are lucky that all final tables are in BCNF. This is not true in general.