

Correction to the learning objective on slide 73 entitled:

**Dependency Preservation:
Using a sound & complete set of inference rules**

The original statement of this learning objective, on the slides posted on the website for CS386, is as follows:

Dependency preservation requires the use of a sound and complete set of rules of inference to compute F^+ , the closure of a set F of FDs. Given the original set of FDs, F . Then G is the set of FDs that are implied by the keys in the resulting decomposition. Dependency preservation is when $F^+ = G^+$.

The sentence that reads “*Then G is the set of FDs that are implied by the keys in the resulting decomposition*” is incorrect. Here’s the correct statement.

Let G consist of every FD in F^+ where all of the attributes involved in the FD appear within a single relation in the normalized schema (after the decomposition).

If I insert the correct statement for this learning objective then it becomes:

Dependency preservation requires the use of a sound and complete set of rules of inference to compute F^+ , the closure of a set F of FDs. Let G consist of every FD in F^+ where all of the attributes involved in the FD appear within a single relation in the normalized schema (after the decomposition). Dependency preservation is when $F^+ = G^+$.

Summary sheet for Normalization

When we normalize a set of relations based on a set of FDs, we have one requirement and two goals.

Requirement: The decomposition must be *lossless*. This means that we don't lose any of the original information that was present in the original relations. Said in another way, we need to make sure that we could easily reproduce the original relations exactly as they appeared before we decomposed any of the relations. We say that the decomposition must be "lossless" (as opposed to "lossy" which is the term used when a decomposition is not lossless). A two-way decomposition based on an FD of $X \rightarrow Y$ (where X and Y are sets of attribute names) is lossless provided the attributes in X serve as a key for at least one of the two relations resulting from the decomposition. Fortunately, our algorithm for "lifting troublesome FDs" guarantees that the decomposition is lossless.

Goal: We would like for all relations to be in *BCNF*. If a relation is in BCNF, this means that there are no FDs present in any of the relations that are NOT implied by the key(s) for that relation. This condition tells us that by simply enforcing the key(s) for this relation, then all FDs present in this relation will be enforced. We also know that there are no update anomalies present based on "troublesome" FDs because there aren't any "troublesome" FDs left.

Goal: We would like for the decomposition, i.e., the resulting schema, to be *dependency-preserving*. This means that all of the original FDs in F (and thus F^+ , all of the FDs derivable from the original set of FDs F) are derivable from the set of all FDs (taken from F^+) that appear within the relations in the schema after we have normalized. An FD (from F^+) appears within a relation R if all attributes in the FD are in R.

Punch line 1: If every relation in the final schema is in BCNF and the decomposition is dependency-preserving, then we know that by enforcing the keys of the relations (which the DBMS will do automatically), then all FDs in the original set are being enforced.

Punch line 2: We can't always decompose a set of relations into BCNF and also preserve all of the FDs. (See a counterexample in the slides associated with the top learning objective in the strand map.) In this case, the DB designer must choose to either decompose all the way to BCNF (and end up with one or more FDs split across two or more relations) or to stop at 3NF for one or more relations (and end up with a bit of redundancy and associated update anomalies in those relations).