**Lossless-join Decomposition**

If \( R \) is a relation schema and \( F \) is a set of FDs over \( R \), a decomposition of \( R \) into 2 schemas with attribute sets \( X \) & \( Y \) is said to be lossless-join if for every instance \( r \) of \( R \) that satisfies the dependencies in \( F \), a natural join of \( X \) & \( Y \) will produce \( r \).

Let’s go back to our schema from our initial example:

\[
\text{Student (StudentID, Name, Major code, Major name).}
\]

We saw in the original example the problems with the redundancy caused by this schema. What if we decompose this into the following schemas:

\[
\begin{align*}
\text{Student (StudentID, Name, Major code)} \\
\text{Major (Major code, Major name)}
\end{align*}
\]

This decomposition is considered a lossless-join decomposition because a natural join between Student and Major will result in the original schema.

**Dependency -preserving**

A decomposition is considered to be dependency preserving if after the decomposition, the dependencies can still be enforced. Let’s look at our contracts example:

\[
\begin{align*}
C & \rightarrow CSJDPQV, JP \rightarrow C, SD \rightarrow P.
\end{align*}
\]

If we decompose the original schema into:

\[
\begin{align*}
(CSJDPV) \text{ and } (SDP)
\end{align*}
\]

We still have a lossless-join decomposition, because a natural join between the two will return the original – however this decomposition has lost the ability to track \( JP \rightarrow C \). \( J \) ended in the first and \( P \) in the second.

Any relation can be decomposed into a 3NF form and both be lossless-join and dependency preserving, however the same cannot be said for BCNF. Given the following scenario:

\[
\begin{align*}
R & = (A,B,C), \quad F = (AB \rightarrow C, \ C \rightarrow A)
\end{align*}
\]

\( R \) is not in BCNF, and you cannot decompose \( R \) into BCNF form and preserve the dependencies; however it is in 3NF.