# Normal Forms

<table>
<thead>
<tr>
<th>Normal form</th>
<th>Defined by</th>
<th>Brief definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>First normal form (1NF)</td>
<td>Two versions: E.F. Codd (1970), C.J. Date (2003)[9]</td>
<td>Table faithfully represents a relation and has no repeating groups</td>
</tr>
<tr>
<td>Second normal form (2NF)</td>
<td>E.F. Codd (1971)[2]</td>
<td>No non-prime attribute in the table is functionally dependent on a proper subset of any candidate key</td>
</tr>
<tr>
<td>Third normal form (3NF)</td>
<td>E.F. Codd (1971)[2]; see also Carlo Zaniolo's equivalent but differently expressed definition (1982)[10]</td>
<td>Every non-prime attribute is non-transitively dependent on every candidate key in the table. The attributes that do not contribute to the description of the primary key are removed from the table. In other words, no transitivity dependency is allowed.</td>
</tr>
<tr>
<td>Elementary Key Normal Form</td>
<td>C.Zaniolo (1982)[10]</td>
<td>Every non-trivial functional dependency in the table is either the dependency of an elementary key attribute or a dependency on a superkey</td>
</tr>
<tr>
<td>Boyce–Codd normal form</td>
<td>Raymond F. Boyce and E.F. Codd (1974)[11]</td>
<td>Every non-trivial functional dependency in the table is a dependency on a superkey</td>
</tr>
<tr>
<td>Fourth normal form (4NF)</td>
<td>Ronald Fagin (1977)[12]</td>
<td>Every non-trivial multivalued dependency in the table is a dependency on a superkey</td>
</tr>
<tr>
<td>Fifth normal form (5NF)</td>
<td>Ronald Fagin (1979)[13]</td>
<td>Every non-trivial join dependency in the table is implied by the superkeys of the table</td>
</tr>
<tr>
<td>Domain/key normal form (DKNF)</td>
<td>Ronald Fagin (1981)[14]</td>
<td>Every constraint on the table is a logical consequence of the table's domain constraints and key constraints</td>
</tr>
<tr>
<td>Sixth normal form (6NF)</td>
<td>C.J. Date, Hugh Darwen, and Nikos Lorentzos (2002)[15]</td>
<td>Table features no non-trivial join dependencies at all (with reference to generalized join operator)</td>
</tr>
</tbody>
</table>
**Minimal Cover**

**Minimal cover** $G$ for a set of FDs $F$:

- Closure of $F$ = closure of $G$. ($F^+ = G^+$)
- Right hand side of each FD in $G$ is a single attribute.
- If we modify $G$ by deleting an FD or by deleting attributes from an FD in $G$, the closure changes.

Intuitively, every FD in $G$ is needed, and ``as small as possible” in order to get the same closure as $F$.

E.g., $A \rightarrow B$, $ABCD \rightarrow E$, $EF \rightarrow GH$, $ACDF \rightarrow EG$ has the following minimal cover:

- $A \rightarrow B$, $ACD \rightarrow E$, $EF \rightarrow G$ and $EF \rightarrow H$
Finding the Minimal Cover

Step 1: From set F, create another set G that has only single attributes in the RHS.

Step 2: Try to find FDs in G that can be removed (one-by-one) and still the new G is equivalent to F (and the original G).

Step 3: Try to remove attributes from LHS for each FD in G such that the new G after removal of the attribute is equivalent to F (and original G). Go back to step 2 and then 3 until stabilize.

Step 4: Put together all the FDs with the same LHS to minimize the number of FDs.
Example for MC algorithm

Let R(A,B,C,D,E,F) and F={ ABD $\rightarrow$ AC, C $\rightarrow$ BE, AD $\rightarrow$ BF, B $\rightarrow$ E}

1) $G = \{ ABD \rightarrow A, ABD \rightarrow C, C \rightarrow B, C \rightarrow E, AD \rightarrow B, AD \rightarrow F, B \rightarrow E\}$

2) -ABD$\rightarrow$ A, trivial, remove it. $G = G - \{ ABD \rightarrow A\}$
   - how about ABD $\rightarrow$ C? No, we cannot remove it. No way to produce this FD from the others.
   - what about C $\rightarrow$ B? We compute C+ in $G - \{ C \rightarrow B\}$: C+ = {CE} so we cannot remove it.
   - What about C$\rightarrow$E? This we can remove! We have C$\rightarrow$B and B$\rightarrow$E that can imply C$\rightarrow$E. So, $G = G - \{ C \rightarrow E\}$
   - all rest essential.

$G = \{ ABD \rightarrow C, C \rightarrow B, AD \rightarrow B, AD \rightarrow F, B \rightarrow E\}$
3) Now, we try to remove single attributes.

- Can we drop A from \( ABD \rightarrow C \) (and create \( G' \))? Is \( G^+ = G'^+ \)?
  We have \( G = \{ ABD \rightarrow C, C \rightarrow B, AD \rightarrow B, AD \rightarrow F, B \rightarrow E \} \) and
  \( G' = \{ BD \rightarrow C, C \rightarrow B, AD \rightarrow B, AD \rightarrow F, B \rightarrow E \} \)

Can we infer \( BD \rightarrow C \) from \( G \)? We compute: \( BD^+ \) in \( G \): \( BD^+ = \{BDE\} \). No. So, not equivalent and cannot remove A.

- What about B in \( ABD \rightarrow C \)? Again, we need to see if we generate \( AD \rightarrow C \) from \( G \).
  We have: \( AD^+ = \{ADBCE\} \). Yes!! So, we can replace \( G \) with \( G' \) =>

  \( G = \{ AD \rightarrow C, C \rightarrow B, AD \rightarrow B, AD \rightarrow F, B \rightarrow E \} \)

- We test the other ones but we cannot remove more attributes.

Go again to Step 2!! => we can remove \( AD \rightarrow B \) now!!
So, we have: $G = \{ \text{AD} \rightarrow \text{C}, \text{C} \rightarrow \text{B}, \text{AD} \rightarrow \text{F}, \text{B} \rightarrow \text{E} \}$

Cannot remove anything else after that.

4) Finally, $G = \{ \text{AD} \rightarrow \text{CF}, \text{C} \rightarrow \text{B}, \text{B} \rightarrow \text{E} \}$ (minimal cover)
Boyce-Codd Normal Form (BCNF)

- Reln R with FDs $F$ is in BCNF if, for all $X \rightarrow A$ in $F^+$
  - $A \in X$ (called a trivial FD), or
  - $X$ is a superkey for $R$.

- In other words: “$R$ is in BCNF if the only non-trivial FDs over $R$ are key constraints.”

- If $R$ in BCNF, then every field of every tuple records information that cannot be inferred using FDs alone.
  - Say we are told that FD $X \rightarrow A$ holds for this example relation:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y1</td>
<td>a</td>
</tr>
<tr>
<td>x</td>
<td>y2</td>
<td>?</td>
</tr>
</tbody>
</table>

- Can you guess the value of the missing attribute?

- Yes, so relation is not in BCNF
Lossless Decomposition

 Decomposition of R into X and Y is lossless-join w.r.t. a set of FDs F if, for every instance \( r \) that satisfies F:

\[
\pi_X(r) \bowtie \pi_Y(r) = r
\]

The decomposition of R into X and Y is lossless with respect to F if and only if \( F^+ \) contains:

\[
X \cap Y \rightarrow X, \quad \text{or} \quad X \cap Y \rightarrow Y
\]

in previous example: decomposing ABC into AB and BC is lossy, because intersection (i.e., “B”) is not a key of either resulting relation.

Useful result: If \( W \rightarrow Z \) holds over R and \( W \cap Z \) is empty, then decomposition of R into R-Z and WZ is lossless.
Lossless Decomposition (example)

But, now we can’t check \( A \rightarrow B \) without doing a join!
Dependency Preserving Decomposition

- Dependency preserving decomposition (Intuitive):

  - If R is decomposed into X, Y and Z, and we enforce the FDs that hold individually on X, on Y and on Z, then all FDs that were given to hold on R must also hold. *(Avoids Problem #2 on our list.)*

- The projection of F on attribute set X (denoted $F_X$) is the set of FDs $U \rightarrow V$ in $F^+$ *(closure of F, not just F)* such that all of the attributes on both sides of the f.d. are in X.

  - That is: $U$ and $V$ are subsets of X
Dependency Preserving Decompositions (Contd.)

- Decomposition of R into X and Y is dependency preserving if
  \[(F_X \cup F_Y)^+ = F^+\]
  i.e., if we consider only dependencies in the closure \(F^+\) that can be checked in X without considering Y, and in Y without considering X, these imply all dependencies in \(F^+\).

- Important to consider \(F^+\) in this definition:
  - ABC, \(A \rightarrow B\), \(B \rightarrow C\), \(C \rightarrow A\), decomposed into AB and BC.
  - Is this dependency preserving? Is \(C \rightarrow A\) preserved????
    - note: \(F^+\) contains \(F \cup \{A \rightarrow C, B \rightarrow A, C \rightarrow B\}\), so...

- \(F_{AB}\) contains \(A \rightarrow B\) and \(B \rightarrow A\); \(F_{BC}\) contains \(B \rightarrow C\) and \(C \rightarrow B\)

- So, \((F_{AB} \cup F_{BC})^+\) contains \(C \rightarrow A\)
Decomposition into BCNF

Consider relation R with FDs F.

If \( X \rightarrow Y \) violates BCNF, decompose R into \( R - Y \) and \( XY \) (guaranteed to be lossless).

Repeated application of this idea will give us a collection of relations that are in BCNF; lossless join decomposition, and guaranteed to terminate.

- e.g., CSJDPQV, key C, JP → C, SD → P, J → S
- \{contractid, supplierid, projectid, deptid, partid, qty, value\}
- To deal with SD → P, decompose into SDP, CSJDQV.
- To deal with J → S, decompose CSJDQV into JS and CJDQV
- So we end up with: SDP, JS, and CJDQV

Note: several dependencies may cause violation of BCNF. The order in which we fix them could lead to very different sets of relations!
Third Normal Form (3NF)

Reln R with FDs $F$ is in 3NF if, for all $X \rightarrow A$ in $F^+$

- $A \subseteq X$ (called a *trivial* FD), or
- $X$ is a superkey of $R$, or
- $A$ is part of some *candidate* key (not superkey!) for $R$. (sometimes stated as “$A$ is *prime*”)

*Minimality* of a key is crucial in third condition above!

- If $R$ is in BCNF, obviously in 3NF.
- If $R$ is in 3NF, some redundancy is possible. It is a compromise, used when BCNF not achievable (e.g., no “good” decom, or performance considerations).

*Lossless-join, dependency-preserving decomposition of R into a collection of 3NF relations always possible.*
Decomposition into 3NF

- Obviously, the algorithm for lossless join decompression into BCNF can be used to obtain a lossless join decompression into 3NF (typically, can stop earlier) but does not ensure dependency preservation.

- To ensure dependency preservation (DP), one idea:
  - If $X \rightarrow Y$ is not preserved, add relation $XY$.

  Problem is that $XY$ may violate 3NF! e.g., consider the addition of CJP to `preserve’ $JP \rightarrow C$. What if we also have $J \rightarrow C$?

- Refinement: Instead of the given set of FDs $F$, use a minimal cover for $F$.

- We can prove that if we use minimal cover in $F$ to ensure DP, then every new relation $XY$ that we add is always in 3NF!
Let R = (A,B,C,D,E,F)

Is R in BCNF? Is R in 3NF?

Since G = {AD→CF, C → B, B → E} is the minimal cover for R.

keys: AD

Not in BCNF, not in 3NF.

Decompose in 3NF and DP:

ACDEF, CB => ABCDF, CB, BE
Another example:

- Relation: CSJDPQV and
  \[ F = \{C \rightarrow CSJDPQV, \ J P \rightarrow C, \ SD \rightarrow P, \ J \rightarrow S\} \]

- $\Rightarrow$ CSJDQV and SDP (but problem with $J \rightarrow S$)
- $\Rightarrow$ CJDQV, SDP, and JS (but what about $JP \rightarrow C$?)
- $\Rightarrow$ add another relation: CJDQV, SDP, JS, JPC