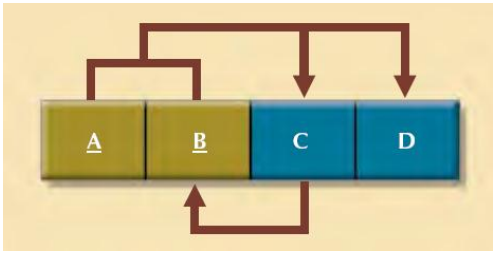
Boyce-Codd Normal Form (BCNF):

A table is in **Boyce-Codd normal form** (**BCNF**) when every determinant in the table is a candidate key.

That a candidate key has the same characteristics as a primary key, but for some reason, it was not chosen to be the primary key. Clearly, when a table contains only one candidate key, the 3NF and the BCNF are equivalent. Putting that proposition another way, BCNF can be violated only when the table contains more than one candidate key.

A table is in 3NF when it is in 2NF and there are no transitive dependencies. But what about a case in which a nonkey attribute is the determinant of a key attribute? That condition does not violate 3NF, yet it fails to meet the BCNF requirements because BCNF requires that every determinant in the table be a candidate key. The situation just described (a 3NF table that fails to meet BCNF requirements) is shown in the following figure:



A table that is in 3NF but not in BCNF

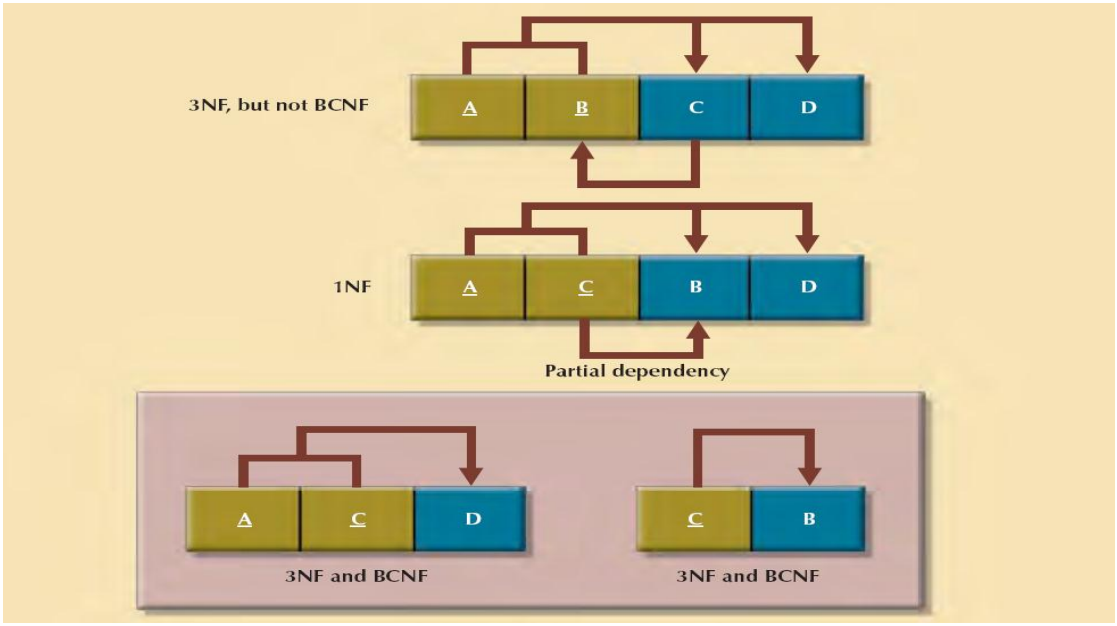
The functional dependencies in the above figure

A + B ---→C, D
A + C --→B, D
C---→B

Notice that this structure has two candidate keys: (A + B) and (A + C). The table structure shown in the above Figure has no partial dependencies, nor does it contain transitive dependencies.

The condition C---→B indicates that *a non key attribute determines part of the primary key*—and *that* dependency is *not* transitive or partial because the dependent is a prime attribute!) Thus, the table structure in the above Figure meets the 3NF requirements. Yet the condition C --→B causes the table to fail to meet the BCNF requirements.
To convert the table structure in the above Figure into table structures that are in 3NF and in BCNF,

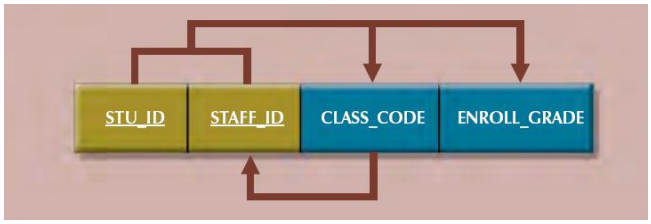
- First change the primary key to A + C. That is an appropriate action because the dependency C -→B means that C is, in effect, a superset of B.
- At this point, the table is in 1NF because it contains a partial dependency, C ---→B. Next, follow the standard decomposition procedures which will follow to produce the 2NF results shown in Figure.



The sample data for the above problem is as shown in the following figure.

STU_ID	STAFF_ID	CLASS_CODE	ENROLL_GRADE
125	25	21334	A
125	20	32456	C
135	20	28458	B
144	25	27563	C
144	20	32456	B

The functional dependencies in the above Relation are as follows:

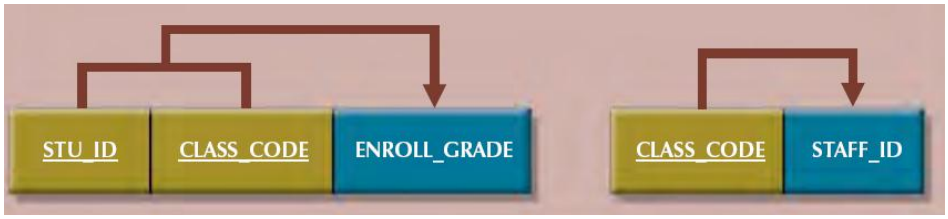


3NF, but not BCNF

The above table reflects the following conditions:

- Each CLASS_CODE identifies a class uniquely. This condition illustrates the case in which a course might generate many classes. For example, a course labeled INFS 420 might be taught in two classes (sections), each identified by a unique code to facilitate registration. Thus, the CLASS_CODE 32456 might identify INFS 420, class section 1, while the CLASS_CODE 32457 might identify INFS 420, class section 2. Or the CLASS_CODE 28458 might identify QM 362, class section 5.
- A student can take many classes. Note, for example, that student 125 has taken both 21334 and 32456, earning the grades A and C, respectively.
- A staff member can teach many classes, but each class is taught by only one staff member. Note that staff member 20 teaches the classes identified as 32456 and 28458.

Table represented by this structure has a major problem, because it is trying to describe two things: staff assignments to classes and student enrollment information. Such a dual-purpose table structure will cause anomalies. For example, if a different staff member is assigned to teach class 32456, two rows will require updates, thus producing an update anomaly. And if student 135 drops class 28458, information about who taught that class is lost, thus producing a deletion anomaly.
The solution to the problem is to decompose the table structure, following the procedure outlined earlier.



3NF and BCNF