ARTICLE 2

Further Notes on Normalization and Higher Normal Forms

Nenad Jukić
Loyola University Chicago

Kornelije Rabuzin
University of Zagreb

The coverage of functional dependencies and normalization given in Chapter 4 of *Database Systems: Introduction to Databases and Data Warehouses (Edition 3.0)* is sufficient for the understanding of the normalization process that occurs in typical corporate and organizational settings. This article gives an extended coverage of functional dependencies and normalization.

Candidate Keys and Functional Dependencies

In addition to the primary key, relations can also have one or more additional candidate keys. Consider the example relation CITY shown in Figure 1. Relation CITY has a primary key CityID and an additional composite candidate key CityName, State. Functional dependencies in the relation CITY are shown in Figure 2.

CITY

CityID	**CityName**	**State**	**StatePopulation**	**CityPopulation**
C1	Portland	ME	1,375,000	70,000
C2	Grand Rapids	MI	10,100,000	190,000
C3	Rockford	IL	12,700,000	340,000
C4	Spokane	WA	7,800,000	210,000
C5	Portland	OR	4,250,000	600,000
C6	Eugene	OR	4,250,000	360,000
C7	Grand Rapids	MN	5,710,000	11,000

Figure 1 Relation CITY with a primary key and a candidate key.

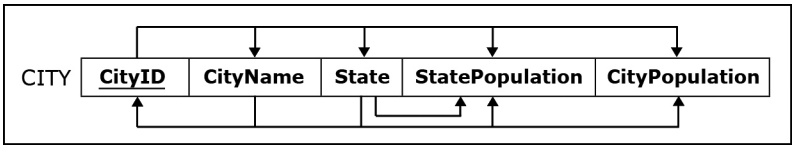

Figure 2 Functional dependencies in the relation CITY.

When a relation has other candidate keys in addition to its primary key, the following expanded definitions of partial and full functional dependencies apply:

Partial Functional Dependency
This occurs when a component of a primary key or any other candidate key on its own functionally determines the non-key columns of a relation.

Full Key Functional Dependency
This occurs when a key (primary or any other candidate key) functionally determines a column of a relation, while no component of the key partially determines the same column.

Note in Figure 2 that the relation CITY has a primary key CityID and a composite candidate key CityName, State. The primary key CityID fully functionally determines all the remaining columns in the relation. Candidate key CityName, State also fully functionally determines all the remaining columns in the relation. The column State functionally determines column StatePopulation. This functional dependency is a partial functional dependency, because column State is a component of a candidate key CityName, State.

Recall the definition of second normal form (2NF):

Second Normal Form (2NF)
A table is in 2NF if it is in first normal form (1NF) and if it does not contain partial functional dependencies.

Relation CITY is not in 2NF because it contains a partial functional dependency. Normalizing relation CITY to 2NF involves eliminating the partial functional dependency by decomposing the relation CITY into two relations, as shown in Figure 3.

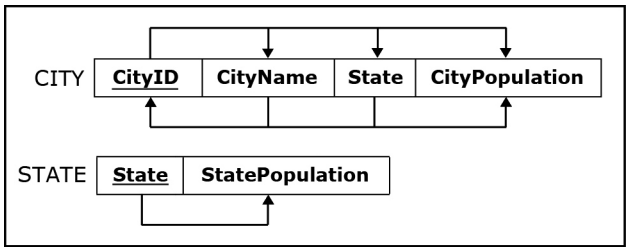

Figure 3 Normalizing relation CITY.

Figure 4 shows the records in the normalized tables.

Recall the definition of a transitive functional dependency:

Transitive Functional Dependency
This occurs when non-key columns functionally determine other non-key columns of a relation.

Also recall the definition of second normal form (3NF):

Third Normal Form (3NF)
A table is in 3NF if it is in 2NF and if it does not contain transitive functional dependencies.

The existence of the candidate key does not call for augmenting the definition of a transitive dependency, because transitive dependency is defined as a dependency between the non-key columns.

The relations in Figure 3 are already in 3NF.

How would the CITY relation in Figure 2 be normalized if we did not acknowledge CityName, State as a candidate key? In that case, the functional dependency

State → StatePopulation

CITY

CityID	CityName	State	CityPopulation
C1	Portland	ME	70,000
C2	Grand Rapids	MI	190,000
C3	Rockford	IL	340,000
C4	Spokane	WA	210,000
C5	Portland	OR	600,000
C6	Eugene	OR	360,000
C7	Grand Rapids	MN	11,000

STATE

State	StatePopulation
ME	1,375,000
MI	10,100,000
IL	12,700,000
WA	7,800,000
OR	4,250,000
MN	5,710,000

Figure 4 Data in the normalized relations for the CITY example.

would be considered a transitive functional dependency. As such, it would be eliminated in the process of normalizing to 3NF instead of being eliminated in the process of normalizing to 2NF. The final result of normalization would be exactly the same as in Figure 3. In other words, relation CITY would end up normalized the same whether CityName, State was treated as a key or not.

Boyce-Codd Normal Form (BCNF)

Boyce-Codd normal form (BCNF) is an extension of the third normal form (3NF). The following is a definition of BCNF:

Boyce-Codd Normal Form (BCNF)
A table is in BCNF if it contains no functional dependencies other than full key functional dependencies (where only the primary key or other candidate keys fully determine other columns).

In most cases, relations that are in 3NF are also in BCNF. A 3NF relation that has no candidate keys other than the primary key is by definition in BCNF. However, there are cases when relations that have candidate keys in addition to the primary keys may be in 3NF while not being in BCNF. For example, consider the relation TEAMPLAYEROFTHEGAME shown in Figure 5. The data in this relation records a team's player of the game for each game in which the team played. In this example, the table contains data for a single season, there are no trades of players (players stay with the same team for the entire season), and no two players have the same name (player's names are unique.) Functional dependencies for the relation TEAMPLAYEROFTHEGAME are shown in Figure 6.

TEAMPLAYEROFTHEGAME

GameOfSeason	Team	TeamPlayerOfTheGame
1st	Tigers	Joe Jones
2nd	Tigers	Tim Smith
3rd	Tigers	Joe Jones
1st	Sharks	Scott McHill
2nd	Sharks	Scott McHill
3rd	Sharks	Lee Hicks

Figure 5 Relation TEAMPLAYEROFTHEGAME (in 3NF but not in BCNF).

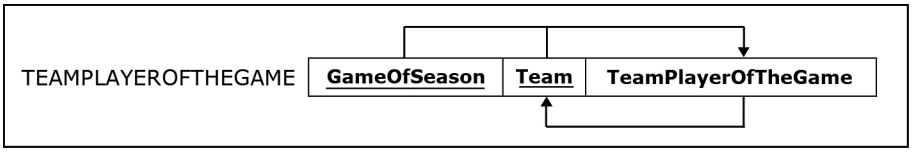

Figure 6 Functional dependencies in relation TEAMPLAYEROFTHEGAME (in 3NF but not in BCNF).

The relation TEAMPLAYEROFTHEGAME is in 3NF because it does not contain any partial or transitive dependencies. However, this relation is not in BCNF because non-key column TeamPlayerOfTheGame determines the key column Team.

The relation TEAMPLAYEROFTHEGAME is normalized to BCNF by creating two relations as shown in Figure 7.

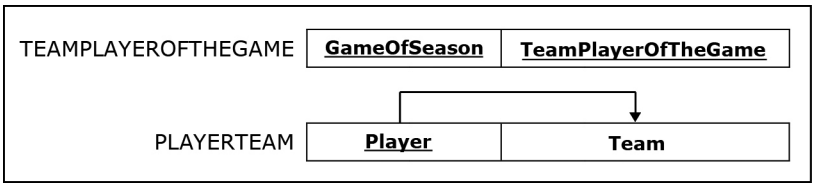

Figure 7 Relation TEAMPLAYEROFTHEGAME normalized to BCNF.

Figure 8 shows the records in the normalized tables.

The issue of normalizing the relation TEAMPLAYEROFTHEGAME to BCNF could have been avoided by choosing a different primary key for the relation. Notice that there are two candidates for the primary key for relation TEAMPLAYEROFTHEGAME shown in Figure 5:

- GameOfSeason, Team
- GameOfSeason, TeamPlayerOfTheGame

In Figure 5, GameOfSeason, Team was chosen to be the primary key of the relation TEAMPLAYEROFTHEGAME. Figure 9 shows the relation TEAMPLAYEROFTHEGAME with the other candidate (GameOfSeason, TeamPlayerOfTheGame) chosen to be its primary key. Functional dependencies for the relation TEAMPLAYEROFTHEGAME are shown in Figure 10.

TEAMPLAYEROFTHEGAME

GameOfSeason	TeamPlayerOfTheGame
1st	Joe Jones
2nd	Tim Smith
3rd	Joe Jones
1st	Scott McHill
2nd	Scott McHill
3rd	Lee Hicks

PLAYERTEAM

Player	Team
Joe Jones	Tigers
Tim Smith	Tigers
Scott McHill	Sharks
Lee Hicks	Sharks

Figure 8 Data in the normalized relations for the TEAMPLAYEROFTHEGAME example.

TEAMPLAYEROFTHEGAME

GameOfSeason	TeamPlayerOfTheGame	Team
1st	Joe Jones	Tigers
2nd	Tim Smith	Tigers
3rd	Joe Jones	Tigers
1st	Scott McHill	Sharks
2nd	Scott McHill	Sharks
3rd	Lee Hicks	Sharks

Figure 9 Relation TEAMPLAYEROFTHEGAME with an alternate primary key.

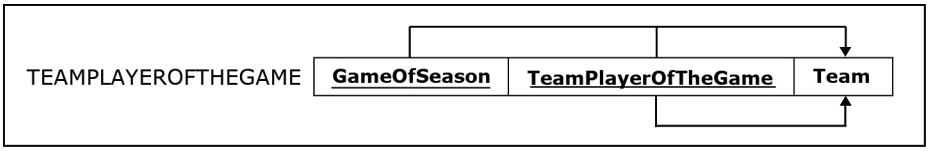

Figure 10 Functional dependencies in relation TEAMPLAYEROFTHEGAME with an alternate primary key.

This version of the relation TEAMPLAYEROFTHEGAME is not in 2NF because it contains a partial dependency. It is normalized to 2NF in a standard way (described in Chapter 4 of *Database Systems: Introduction to Databases and Data Warehouses [Edition 3.0]*) by creating an additional relation for the partial dependency, as shown in Figure 11.

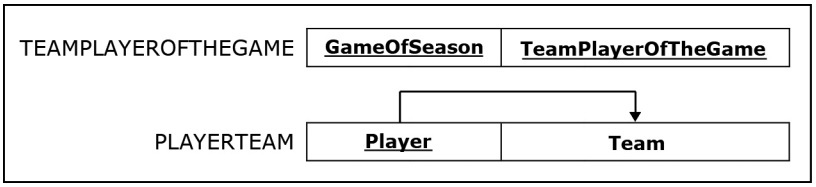

Figure 11 Relation TEAMPLAYEROFTHEGAME with an alternate primary key, normalized to 2NF (and subsequently to 3NF and BCNF).

Note that the relations shown in Figure 11 are also normalized to 3NF and BCNF. In fact, the relations in Figure 11 are identical to the relations in Figure 7. Conversely, the data records for the relations shown in Figure 11 are the same as the data records shown in Figure 8.

In this case, by choosing a different primary key, the normalization process involved dealing with a partial functional dependency instead of dealing with a functional dependency where a non-key attribute determines a key column. In other words, by choosing a different primary key, the normalization process consisted of normalizing to 2NF in order to normalize the relation instead of normalizing to BCNF in order to normalize the relation.

Now let us consider an example when that is not the case. Consider the relation DISPATCHINGTRUCKS shown in Figure 12. The data in this relation records the load of dispatched trucks as well as when they were dispatched and by which dispatcher. Each truck can be dispatched only once a day by one dispatcher. Each day, one or more dispatchers are dispatching trucks, and we record how many hours each dispatcher worked on each day. Functional dependencies for the relation DISPATCHINGTRUCKS are shown in Figure 13.

DISPATCHINGTRUCKS

Date	TruckID	Dispatcher	Dhours	Load
1.1.2025.	T1	Disp1	2	0.5 tons
1.1.2025.	T2	Disp2	3	1.2 tons
1.1.2025.	T3	Disp2	3	1.5 tons
1.2.2025.	T1	Disp2	6	2.0 tons
1.2.2025.	T2	Disp2	6	2.1 tons
1.2.2025.	T3	Disp2	6	1.7 tons

Figure 12 Relation DISPATCHINGTRUCKS (in 3NF but not in BCNF).

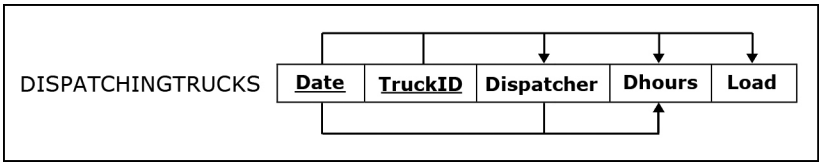

Figure 13 Functional dependencies in relation DISPATCHINGTRUCKS (in 3NF but not in BCNF).

The relation DISPATCHINGTRUCKS is in 3NF because it does not contain any partial or transitive dependencies. However, this relation is not in BCNF because combination of the non-key column Dispatcher and a key column Date determines the non-key column Dhours.

The relation DISPATCHINGTRUCKS is normalized to BCNF by creating two relations as shown in Figure 14.

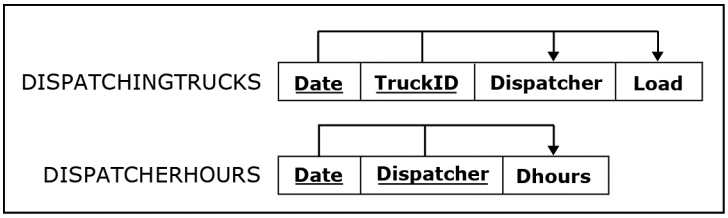

Figure 14 Relation DISPATCHINGTRUCKS normalized to BCNF.

Figure 15 shows the records in the normalized tables.

DISPATCHINGTRUCKS

Date	TruckID	Dispatcher	Load
1.1.2025.	T1	Disp1	0.5 tons
1.1.2025.	T2	Disp2	1.2 tons
1.1.2025.	T3	Disp2	1.5 tons
1.2.2025.	T1	Disp2	2.0 tons
1.2.2025.	T2	Disp2	2.1 tons
1.2.2025.	T3	Disp2	1.7 tons

DISPATCHERHOURS

Date	Dispatcher	Dhours
1.1.2025.	Disp1	2
1.1.2025.	Disp2	3
1.2.2025.	Disp2	6

Figure 15 Data in the normalized relations for the DISPATCHINGTRUCKS example.

Fourth Normal Form (4NF)

Consider the relation ORGANIZATION_STUDENT_CHARITY in Figure 16.

In this example, organizations have student members and organizations support various charities. The relation ORGANIZATION_STUDENT_CHARITY shows for each organization both its members and the charities the organization supports. Columns StudentID and Charity are both related to the column OrgID, but they are not related to each other. One OrgID value can be associated with multiple StudentID values, and separately one OrgID value can be associated with multiple charity values. Formally this is depicted as

ORGANIZATION_STUDENT_CHARITY

OrgID	StudentID	Charity
O11	1111	Food Pantry
O11	1111	Stop Diabetes
O11	1111	River Care
O22	1111	River Care
O22	2222	River Care

Figure 16 Relation ORGANIZATION_STUDENT_CHARITY (not in 4NF).

$$\text{OrgID} \twoheadrightarrow \text{StudentID}$$
$$\text{OrgID} \twoheadrightarrow \text{Charity}$$

where the symbol ↠ (double arrow) indicates a multivalued dependency. **Multivalued dependencies** are often referred to as "tuple (row) generating dependencies." For example, because of OrgID ↠ Charity, every time a new student joins an organization, several new rows have to be generated (one for each charity that the organization is associated with). Also, because of OrgID ↠ StudentID, every time a new charity becomes associated with an organization, several new rows have to be generated (one for each student member of the organization).

As we illustrated, multivalued dependencies occur when separate columns of the same relation contain unrelated values (i.e., when the same relation represents separate relationships of cardinality greater than 1). The following is a definition of **fourth normal form (4NF)**:

ORGANIZATION_STUDENT

OrgID	StudentID
O11	1111
O22	1111
O22	2222

ORGANIZATION_CHARITY

OrgID	Charity
O11	Food Pantry
O11	Stop Diabetes
O11	River Care
O22	River Care

Figure 17 Relation ORGANIZATION_STUDENT_CHARITY normalized to 4NF.

Fourth Normal Form (4NF)
A table is in 4NF if it is in BCNF and does not contain multivalued dependencies.

Consequently, a relation is not in 4NF if it contains multivalued dependencies.

Because of the multivalued dependencies, the relation ORGANIZATION_STUDENT_CHARITY is not in 4NF. Normalizing to 4NF simply involves creating a separate relation for each of the multivalued dependencies as shown in Figure 17.

In most cases, in practice, relations that are in 3NF are also already in 4NF. For example, most designers modeling the organization depicted in the previous example would immediately create the two relations shown in Figure 17 that are in 3NF and 4NF. Only a particularly artificial and/or inept effort would result in creating a table shown in Figure 16 that is in 3NF and then normalizing it to 4NF as shown in Figure 17.

Other Normal Forms

In addition to 4NF, there are other higher normal forms, such as fifth normal form (5NF) and domain key normal form (DKNF). Such normal forms use theoretical concepts that are rarely encountered in practice. Consequently, these concepts are beyond the scope of this book.

Note About the Term "Partial Functional Dependency"

In this book we use a simplified version of the term "partial functional dependency." For example, assume that A, B is a composite key in the relation with columns A, B, C. If, in this relation, B functionally determines C, in this book, we would state that B → C is a partial functional dependency. Technically, according to the original definition of partial dependency, B → C is causing the A, B → C dependency to be a partial dependency. Either way, we still need to move B → C to another table in order to eliminate the partial dependency (and normalize to 2NF). We chose to use the simpler version of the term "partial functional dependency" for brevity and ease of reading. No original meaning is lost using the simplified version. In both versions, the partial dependency occurs when a non-key attribute is determined by a part of the primary key.