Chapter 7. Normalization

First Normal Form (1NF)

- In the relational model, we formalize this idea that attributes do not have any sub-structure.
- A domain is atomic if elements of the domain are considered to be indivisible units.
- We say that a relation schema R is in first normal form (1NF) if
  - the domains of all attributes of R are atomic.

- For example, if the schema of a relation *employee* included an attribute *children* whose domain elements are sets of names, the schema would not be in first normal form.
  - Multivalued attribute
- Composite attributes, such as an attribute *address* with component attributes *street* and *city* also have non-atomic domains.

Second Normal Form (2NF)

- A functional dependency $\alpha \rightarrow \beta$ is called a partial dependency if there is a proper subset γ ($\gamma \subset \alpha$) of α such that $\gamma \rightarrow \beta$; we say that β is partially dependent on α.
- A relation schema *R* is in second normal form (2NF) if each attribute *A* in *R* meets one of the following criteria:
  - It appears in a candidate key.
  - It is not partially dependent on a candidate key.

Another Equivalent Definition

- A relation is in the second normal form if it fulfills the following two requirements:
  - It is in first normal form.
  - It does not have any non-prime attribute that is functionally dependent on any proper subset of any candidate key of the relation.
    - A non-prime attribute of a relation is an attribute that is not a part of any candidate key of the relation.

https://en.wikipedia.org/wiki/Second_normal_form

- Put simply, a relation is in 2NF if
  - it is in 1NF and
  - every non-prime attribute of the relation is dependent on the whole of every candidate key.
- Second normal form is of only historical interest since it provides no benefit over 3NF.
- Every 3NF schema is in 2NF.

https://en.wikipedia.org/wiki/Second_normal_form

Boyce-Codd Normal Form (BCNF)

- A relation schema R is in BCNF with respect to a set F of functional dependencies if for all functional dependencies in F^+ of the form $\alpha \rightarrow \beta$ where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R
- BCNF is also called 3.5NF

- It is always possible to decompose a relation into a set of relations that are in BCNF such that:
  - The decomposition is lossless
  - It may not be possible to preserve dependencies.
    - Checking whether *F* holds on *R* is inefficient since a join is necessary.

Third Normal Form (3NF)

- A relation schema R is in third normal form (3NF) if for all $\alpha \rightarrow \beta$ in F^+ at least one of the following holds:
- $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
- α is a superkey for R
- Each attribute A in $\beta - \alpha$ is contained in a candidate key for R.

Original Definition

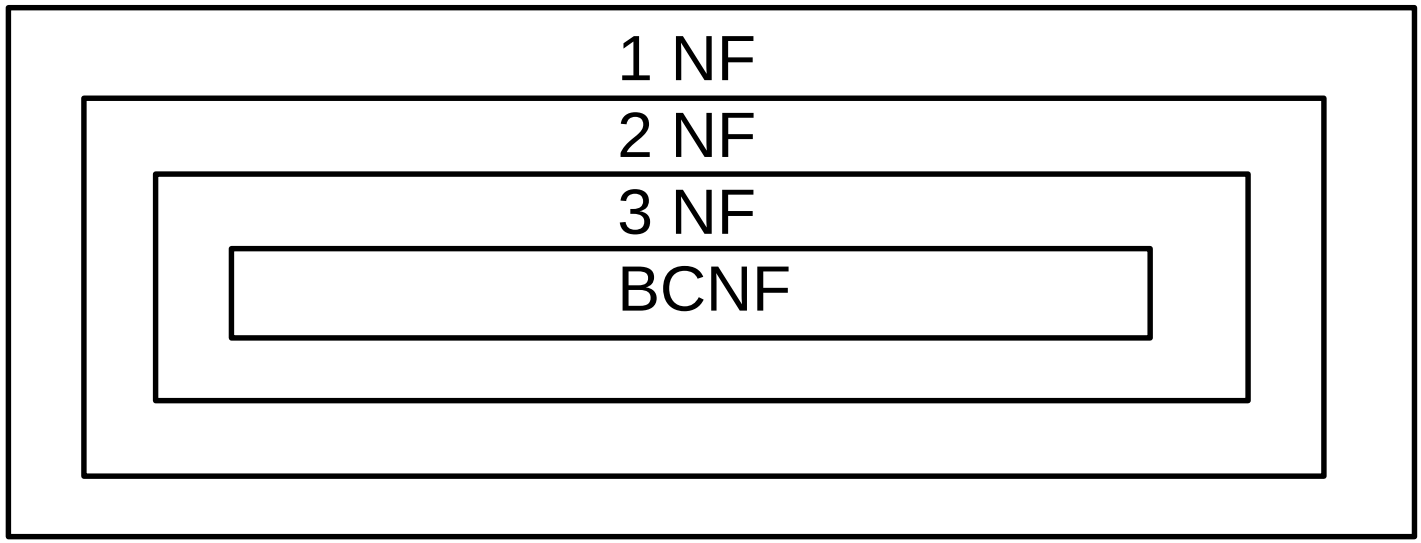
- A table is in 3NF if and only if both of the following conditions hold:
  - The relation *R* (table) is in second normal form (2NF).
  - Every non-prime attribute of *R* is non-transitively dependent on every key of *R*.

https://en.wikipedia.org/wiki/Third_normal_form

- A non-prime attribute of *R* is an attribute that does not belong to any candidate key of *R*.
- A transitive dependency is a functional dependency in which X → Z (X determines Z) indirectly, by virtue of X → Y and Y → Z (where it is not the case that Y → X).

- It is always possible to decompose a relation into a set of relations that are in 3NF such that:
  - The decomposition is lossless
  - The dependencies are preserved
- Disadvantage of 3NF over BCNF
  - Repetition of information (redundancy)
  - Null value

- To test for BCNF
- To test decomposition for BCNF
- To compute BCNF decomposition
- To test for 3NF
  - NP-hard
- To compute 3NF decomposition


1 NF
2 NF
3 NF
BCNF

Un-normalized Table

- Violate 1NF since *Telephone Number* is not atomic

Customer

Customer ID	First Name	Surname	Telephone Number
123	Pooja	Singh	555-861-2025, 192-122-1111
456	San	Zhang	(555) 403-1659 Ext. 53; 182-929-2929
789	John	Doe	555-808-9633

https://en.wikipedia.org/wiki/First_normal_form

Making IT Conform to 1NF

- Method 1: Flatten the table
  - Problem: Result in repeated values

Customer

Customer ID	First Name	Surname	Telephone Number
123	Pooja	Singh	555-861-2025
123	Pooja	Singh	192-122-1111
456	San	Zhang	182-929-2929
456	San	Zhang	(555) 403-1659 Ext. 53
789	John	Doe	555-808-9633

- Method 2: Create a new table
  - *Telephone Number ID* is not necessary

Customer Name

Customer ID	First Name	Surname
123	Pooja	Singh
456	San	Zhang
789	John	Doe

Customer Telephone Number

Telephone Number ID	Customer ID	Telephone Number
1	123	555-861-2025
2	123	192-122-1111
3	456	(555) 403-1659 Ext. 53
4	456	182-929-2929
5	789	555-808-9633

2NF

- An example table violates 2NF

Electric toothbrush models

Manufacturer	Model	Model full name	Manufacturer country
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Brushmaster	SuperBrush	Brushmaster SuperBrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaster	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany

https://en.wikipedia.org/wiki/Second_normal_form

- The above relation *does not* satisfy 2NF because:
  - {*Manufacturer country*} is functionally dependent on {*Manufacturer*}
    - *Manufacturer* → *Manufacturer country*
  - {*Manufacturer country*} is not part of a candidate key, so it is a non-prime attribute
  - {*Manufacturer*} is a proper subset of {*Manufacturer, Model*} candidate key
  - Since {*Manufacturer country*} is a non-prime attribute functionally dependent on a part of a candidate key, the relation is in violation of 2NF.

Making It Conform to 2NF

Electric toothbrush manufacturers

Manufacturer	Manufacturer country
Forte	Italy
Dent-o-Fresh	USA
Brushmaster	USA
Kobayashi	Japan
Hoch	Germany

Electric toothbrush models

Manufacturer	Model	Model full name
Forte	X-Prime	Forte X-Prime
Forte	Ultraclean	Forte Ultraclean
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush
Brushmaster	SuperBrush	Brushmaster SuperBrush
Kobayashi	ST-60	Kobayashi ST-60
Hoch	Toothmaster	Hoch Toothmaster
Hoch	X-Prime	Hoch X-Prime

3NF

- An example of a 2NF table that fails to meet the requirements of 3NF is:

Tournament winners

Tournament	Year	Winner	Winner's date of birth
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

https://en.wikipedia.org/wiki/Third_normal_form

- {*Tournament, Year*} is a candidate key for the table.
- Functional dependencies
  - *Tournament*, *Year* → *Winner*
  - *Winner* → *Date of birth*
- According to the definition in the textbook, this table violates 3NF due to the following reasons.
  - *Winner* is not a superkey
  - *Winner* is not contained in a candidate key

- According to the original definition of 3NF, the violation of 3NF occurs because
  - the non-prime attribute *Date of birth* is transitively dependent on the candidate key {*Tournament, Year*} through the non-prime attribute *Winner*.

Making It Conform to 3NF

Tournament winners

Tournament	Year	Winner
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

Winner's dates of birth

Winner	Date of birth
Chip Masterson	14 March 1977
Al Fredrickson	21 July 1975
Bob Albertson	28 September 1968

Algorithms

- To compute closure of functional dependencies (i.e., F^+)
- To compute closure of attribute sets
  - Attribute closure can be used to
    - test for superkey,
    - test functional dependencies, and
    - compute closure of functional dependencies *F*
- To test whether an attribute is extraneous
- To compute canonical cover

- To test dependency preservation
  - Use closure of function dependencies
    - Need to compute F^+.
    - High complexity
  - Use attribute closure
- To test if a relation is in BCNF
  - Using only F is enough
- To test if a decomposition is in BCNF
  - Using only F is incorrect
  - Using F^+ is correct

- Testing a given schema to see if it is in 3NF is NP-hard.
- Decomposition a scheme into 3NF can be implemented in polynomial time.