Database Design/ Data Modelling

By –

Rajeev Srivastava

C-DAC Mumbai

Sep, 2019

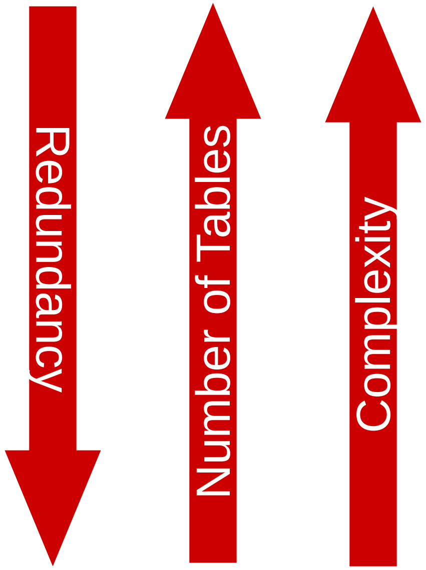
Normalization

Normalization: What & Why

- An analytical technique used during logical database design
- This is the process which allows you to winnow out redundant data within your database.
- This involves restructuring the tables to avoid data inconsistencies (**insert/update/delete anomalies**)
- A properly normalized database should have the following characteristics
  - Scalar/Atomic values in each fields
  - No data redundancy.
  - Minimal/No loss of information
  - Minimal Uses of NULL values

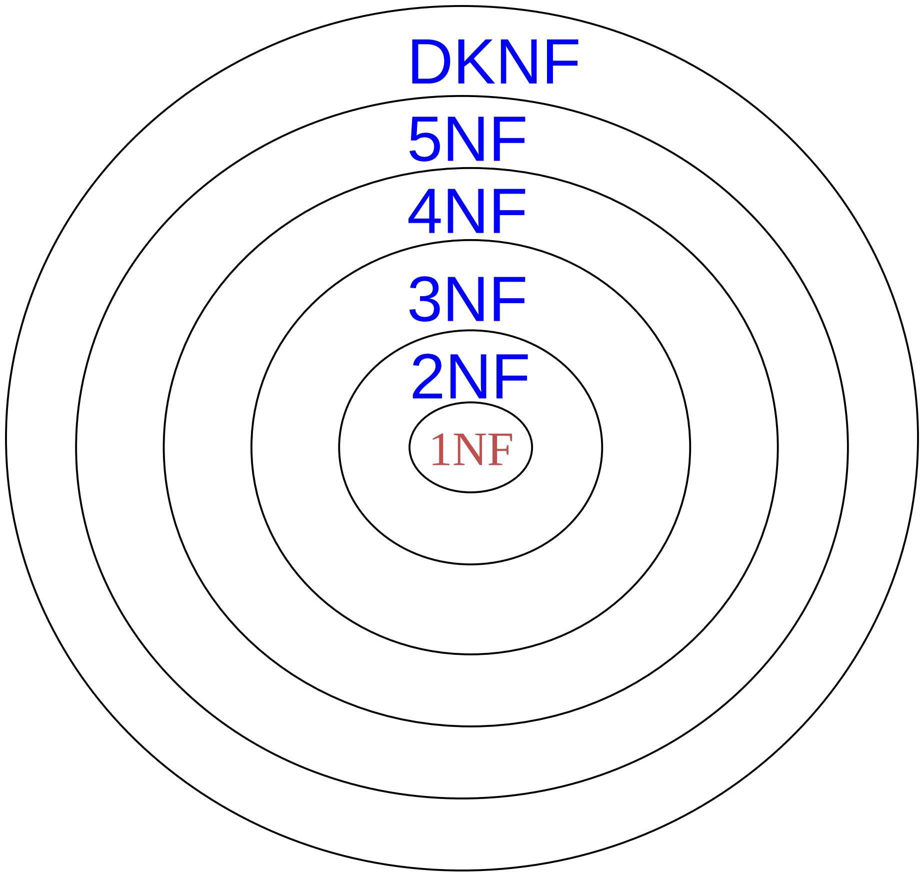
Levels of Normalization

- Levels of normalization based on the amount of redundancy in the database.
- Various levels of normalization are:
    - First Normal Form (1NF)
    - Second Normal Form (2NF)
    - Third Normal Form (3NF)
    - Boyce-Codd Normal Form (BCNF)
    - 4NF & 5 NF
    - 6th NF or Domain Key Normal Form (DKNF)



- Most databases should be in 3NF or BCNF in order to avoid the database anomalies.

Levels of Normalization



Each lower level is a subset of the higher level

Data Anomalies

- INSERT Anomaly
- UPDATE Anomaly
- DELETE Anomaly

Roll No	S.Name	Address	Phone No	CourseID	CourseName	CourseFee
S001	Aditya	Jaipur	90000000	C1	Java	15000
S001	Aditya	Jaipur	90000000	C2	Database	20000
S002	Omkar	Mumbai	70000000	C2	Database	20000
S003	Aniket	Pune	80000000	C1	Java	15000
S004	Shweta	Pune	20000000	C3	OS	18000

First Normal Form (1NF)

A table is considered to be in 1NF if all the fields contain only scalar values (as opposed to list of values).

Example (Not 1NF)

ISBN	**Title**	**AuName**	**AuPhone**	**PubName**	**PubPhone**	**Price**
0-321-32132-1	Core Java	Sleepy, Snoopy, Grumpy	321-321-1111, 232-234-1234, 665-235-6532	Small House	714-000-0000	$34.00
0-55-123456-9	Postgres	Jones, Smith	123-333-3333, 654-223-3455	Small House	714-000-0000	$22.95
0-123-45678-0	Oracle	Joyce	666-666-6666	Alpha Press	999-999-9999	$34.00
1-22-233700-0	MongoDB	Roman	444-444-4444	Big House	123-456-7890	$25.00

Author and AuPhone columns are not scalar

1NF - Decomposition

1. Place all items that appear in the repeating group in a new table
2. In the new table add the primary key column of the table from which the repeating group was extracted.

Other Considerations

- Each row must be unique identifiable.
- Each column must store same type of values

1NF - Decomposition

Example (1NF)

ISBN	Title	PubName	PubPhone	Price
0-321-32132-1	Core Java	Small House	714-000-0000	$34.00
0-55-123456-9	Postgres	Small House	714-000-0000	$22.95
0-123-45678-0	Oracle	Alpha Press	999-999-9999	$34.00
1-22-233700-0	MongoDB	Big House	123-456-7890	$25.00

ISBN	AuName	AuPhone
0-321-32132-1	Sleepy	321-321-1111
0-321-32132-1	Snoopy	232-234-1234
0-321-32132-1	Grumpy	665-235-6532
0-55-123456-9	Jones	123-333-3333
0-55-123456-9	Smith	654-223-3455
0-123-45678-0	Joyce	666-666-6666
1-22-233700-0	Roman	444-444-4444

Functional Dependencies

If one set of attributes in a table determines another set of attributes in the table, then the second set of attributes is said to be functionally dependent on the first set of attributes.

Example 1

ISBN	Title	Price
0-321-32132-1	Core Java	$34.00
0-55-123456-9	Postgres	$22.95
0-123-45678-0	Oracle	$34.00
1-22-233700-0	MongoDB	$25.00

Table Scheme: {ISBN, Title, Price}

Functional Dependencies: {ISBN} → {Title}

{ISBN} → {Price}

Functional Dependencies

Example 2

PubID	PubName	PubPhone
1	Big House	999-999-9999
2	Small House	123-456-7890
3	Alpha Press	111-111-1111

Table Scheme: {PubID, PubName, PubPhone}

Functional Dependencies: {PubId} → {PubPhone}
{PubId} → {PubName}
{PubName, PubPhone} → {PubID}

Example 3

AuID	AuName	AuPhone
1	Sleepy	321-321-1111
2	Snoopy	232-234-1234
3	Grumpy	665-235-6532
4	Jones	123-333-3333
5	Smith	654-223-3455
6	Joyce	666-666-6666
7	Roman	444-444-4444

Table Scheme: {AuID, AuName, AuPhone}

Functional Dependencies: {AuId} → {AuPhone}
{AuId} → {AuName}
{AuName, AuPhone} → {AuID}

Second Normal Form (2NF)

- For a table to be in 2NF, there are two requirements
  - The database is in first normal form
  - All **nonkey** attributes in the table must be fully functionally dependent on the **entire primary key**

- **Example 1 (Not 2NF)**

- **Scheme →** {Title, PubId, AuId, Price, AuAddress}
  - **Key →** {Title, PubId, AuId}
  - {Title, PubId, AuID} → {Price}
  - {AuID} → {AuAddress}
  - **AuAddress does not belong to a key**
  - **AuAddress functionally depends on AuId which is a subset of a key**

Second Normal Form (2NF)

Example 2 (Not 2NF)

Scheme → {City, Street, HouseNumber, HouseColor, CityPopulation}

- **Key** → {City, Street, HouseNumber}
- {City, Street, HouseNumber} → {HouseColor}
- {City} → {CityPopulation}
- CityPopulation does not belong to any key.
- CityPopulation is functionally dependent on the City which is a subset of the key.

2NF - Decomposition

- If a data item is fully functionally dependent on only a part of the primary key, move that data item and that part of the primary key to a new table.
- If other data items are functionally dependent on the same part of the key, place them in the new table also
- Make the partial primary key copied from the original table the primary key for the new table. Place all items that appear in the repeating group in a new table
- **Example 1 (Convert to 2NF)**

Old Scheme → {Title, PubId, AuId, Price, AuAddress}

New Scheme → {Title, PubId, AuId, Price}

New Scheme → {AuId, AuAddress}

2NF - Decomposition

Example 2 (Convert to 2NF)

Old Scheme → {City, Street, HouseNumber, HouseColor, CityPopulation}

New Scheme → {City, Street, HouseNumber, HouseColor}

New Scheme → {City, CityPopulation}

Third Normal Form (3NF)

- This form dictates that all **non-key** attributes must directly depend on key attribute i.e.. **no interdependencies among non-key attributes**.
- For a table to be in 3NF, there are two requirements
  - The table should be second normal form
  - No attribute is **transitively dependent** on the primary key

Third Normal Form (3NF)

Example 1 (Not in 3NF)

Scheme → {Studio, StudioCity, CityTemp}

- Key → {Studio}
- {Studio} → {StudioCity} | {StudioCity} → {CityTemp} | {Studio} → {CityTemp}
- Both StudioCity and CityTemp depend on the entire key hence 2NF
- **CityTemp** transitively depends on **Studio** hence violates 3NF

Example 2 (Not in 3NF)

Scheme → {BuildingID, Contractor, Fee}

- Key → {BuildingID}
- {BuildingID} → {Contractor}
- {Contractor} → {Fee}
- {BuildingID} → {Fee}
- Both Contractor and Fee depend on the entire key (2NF)
- **Fee** transitively depends on the **BuildingID**

BuildingID	**Contractor**	**Fee $**
100	Shiva	1200
150	Akash	1100
200	Shiva	1200
250	Prithvi	1000
300	Shiva	1200

3NF - Decomposition

- Move all items involved in transitive dependencies to a new entity.
- Identify a primary key for the new entity.
- Place the primary key for the new entity as a foreign key on the original entity.

3NF - Decomposition

Example 1 (Convert to 3NF)

Old Scheme → {Studio, StudioCity, CityTemp}

New Scheme → {Studio, StudioCity}

New Scheme → {StudioCity, CityTemp}

Example 2 (Convert to 3NF)

Old Scheme → {BuildingID, Contractor, Fee}

New Scheme → {BuildingID, Contractor}

New Scheme → {Contractor, Fee}

BuildingID	Contractor
100	Shiva
150	Akash
200	Shiva
250	Prithvi
300	Shiva

Contractor	Fee
Shiva	1200
Akash	1100
Prithvi	1000

Boyce-Codd Normal Form (BCNF)

- BCNF does not allow dependencies between attributes that belong to candidate key.
- BCNF is a refinement of the third normal form in which it drops the restriction of a non-key attribute from the 3rd normal form.

Boyce-Codd Normal Form (BCNF)

Example - Movie (Not in BCNF)

Scheme → {MovieID, MovieTitle, ActorName, Payment }

- Key1 → {<u>MovieID, ActorName</u>}
- Key2 → {<u>MovieTitle, ActorName</u>}
- {MovieID} → {MovieTitle}

- Payment functionally depend on both Candidate Keys, thus 3NF
- Dependency between MovieID & MovieTitle (Non-Trivial Dependency) Violates BCNF

BCNF - Decomposition

- Place the two attributes of the non-trivial dependency into a new table.
- Copy the determinant attribute of the non-trivial dependency with all other non-key attributes into a separate table.
- Define primary key in both the tables.

BCNF - Decomposition

Example (Convert to BCNF)

Old Scheme → {MovieID, MovieTitle, ActorName, Payment }

New Scheme → {MovieID, MovieTitle}

New Scheme → {MovieID, ActorName, Payment}

Decomposition – Loss of Information

- **Lossless** decomposition - If decomposition does not cause any loss of information.
- **Dependency-preserving** decomposition - Does not cause any dependencies to be lost.
- **Dependency preservation is not guaranteed** in BCNF decomposition
- Any table can be decomposed in a **lossless** way into **3rd normal form** that **also preserves the dependencies**.
- **3NF may be better than BCNF in some cases**