Database Normalization

Contents

- Functional Dependency
- Normalization

✓ First normal form
✓ Second normal form
✓ Third normal form
✓ BCNF
✓ Fourth normal form
✓ Fifth normal form

along with their problems.

Functional Dependency

- is the association between two set of attributes of a single relation for any relation R having set of attributes (X, Y)

$$X \rightarrow Y$$

1. X functionally determines Y or Y is functionally dependent on X
2. X is the determinant and Y is the dependent.
3. An attribute in a relational model is said to be **functionally dependant** on another attribute in the table if it can take only one value for a given value of the attribute upon which it is functionally dependant.(For a single value of X there is single value of Y.)

S#	CITY	P#	QTY
S1	LONDON	P1	300
S1	LONDON	P2	200
S1	LONDON	P3	400
S1	LONDON	P4	200
S1	LONDON	P5	100
S1	LONDON	P6	100
S2	PARIS	P1	300
S2	PARIS	P2	400
S3	PARIS	P2	200
S4	LONDON	P2	200
S4	LONDON	P4	300
S4	LONDON	P5	400

Possible FDs:

{S#} → {City},

{S#, P#} → {Qty},

{S#, P#} → {City},

{S#, P#} → {City, Qty}

{S#, P#} →{Qty}

{S#, P#} →{S#}

{S#, P#} →{P#}

{S#, P#} →{S#,P#,City,Qty}

{S#} →{Qty}

{S#, P#} → {S#},{Qty} → {S#} and many more

3 Ways to Control FDs

- **Avoid Trivial FDs:**
- Trivial: a) RHS is subset of LHS.
- b) auto satisfied and auto commit
- **Apply Closure Principle:**

Closure: If S is set of FDs then S+ is closure of S if it will include all FDs implied by S after apply 7 Armstrong's axioms:

- **Reflexivity :** if A→B, A determines B where A is subset of B
- **Augmentation :** if A→B, then AC→BC
- **Transitivity :** if A→B and B→C then A→C
- **Self determination :** A→A
- **Union :** if A→B and A→C then AD→BE
- **Composition :** if A→B and D→E then AD→BE
- **Decomposition:** if A→BC then A→B and A→C.

Controlling FDs

- **Apply Irreducibility Principle:**

For any FD:

- RHS is **singleton** set. It means that right side of every FD in S involves just one attribute.
- LHS is irreducible, it means no attribute can be discarded from determinant without changing closure S^+. such an FD is called **left-irreducible.**
- No FD in S can be discarded S without changing closure S^+.

Normalization

It refers to the **decomposing of sets of relations** into desired set of relations having following properties:

- Minimized redundancy
- Minimized update anomalies
- No loss of information
- Maintain functional dependency

- This is the process which allows you to winnow out redundant data within your database.
- This involves restructuring the tables to successively meeting higher forms of Normalization.
- A properly normalized database should have the following characteristics
  - Scalar values in each fields
  - Absence of redundancy.
  - Minimal use of null values.
  - Minimal loss of information.

Normalisation

Is derivation of data as a set of

Non-Redundant,

Consistent and

Inter-Dependent Relations

Normalisation - Advantages

- Reduction of data redundancy within tables
- Reduce data storage space
- Reduce inconsistency of data
- Reduce update cost
- Remove many-to-many relationship
- Improve flexibility of the system

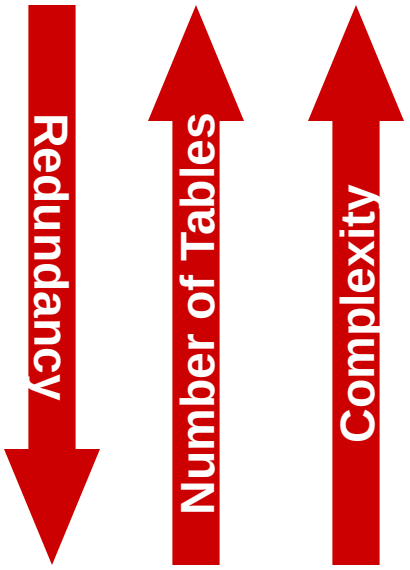
Normalisation - Disadvantages

Reduction in efficiency of certain data retrieval as relations may be joined during retrieval.

- **Increase of joins**
- **Increase use of storage (keys)**
- **Increase complexity of the system**

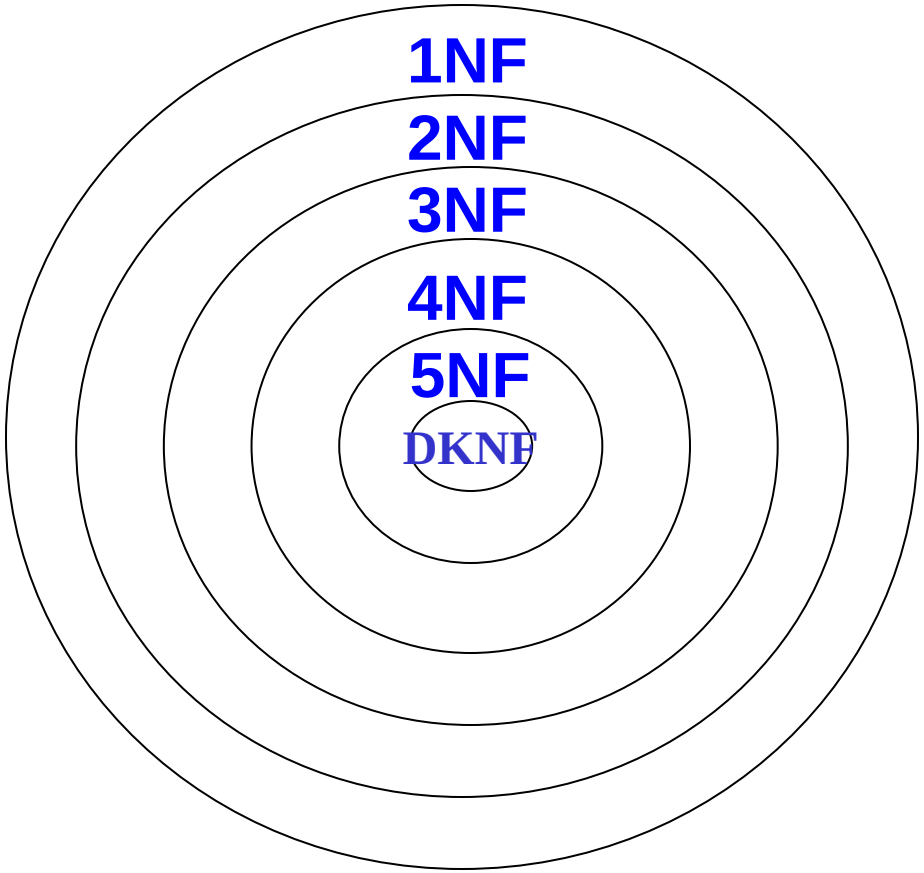
Levels of Normalization

- 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)
  - Fourth Normal Form (4NF)
  - Fifth Normal Form (5NF)
  - Domain Key Normal Form (DKNF)



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

Levels of Normalization



Each higher level is a subset of the lower level

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	Balloon	Sleepy, Snoopy, Grumpy	321-321-1111, 232-234-1234, 665-235-6532	Small House	714-000-0000	$34.00
0-55-123456-9	Main Street	Jones, Smith	123-333-3333, 654-223-3455	Small House	714-000-0000	$22.95
0-123-45678-0	Ulysses	Joyce	666-666-6666	Alpha Press	999-999-9999	$34.00
1-22-233700-0	Visual Basic	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. Designate a primary key for each new table produced.
3. Duplicate in the new table the primary key of the table from which the repeating group was extracted or vice versa.

Example (1NF)

ISBN	Title	PubName	PubPhone	Price
0-321-32132-1	Balloon	Small House	714-000-0000	$34.00
0-55-123456-9	Main Street	Small House	714-000-0000	$22.95
0-123-45678-0	Ulysses	Alpha Press	999-999-9999	$34.00
1-22-233700-0	Visual Basic	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

Second Normal Form

- **Second normal form** (**2NF**) If a table has a composite key, all attributes must be related to the whole key:
- The database must meet all the requirements of the first normal form.
- Data which is redundantly duplicated across multiple rows of a table is moved out to a separate table.

Second Normal Form (2NF)

All **Non Key** attributes in the table must be functionally dependent on the entire primary key

Note: *Remember that we are dealing with non-key attributes*

Example (Not 2NF)

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

1. Key → {Title, PubId, AuId}
2. {Title, PubId, AuID} → {Price}
3. {AuID} → {AuAddress}
4. AuAddress does not belong to a key
5. AuAddress functionally depends on AuId which is a subset of a key

2NF - Decomposition

1. 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.
2. If other data items are functionally dependent on the same part of the key, place them in the new table also
3. 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 (Convert to 2NF)

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

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

New Scheme → {AuId, AuAddress}

Second Normal Form (2NF)

Example (Not 2NF)

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

1. key → {City, Street, HouseNumber}
2. {City, Street, HouseNumber} → {HouseColor}
3. {City} → {CityPopulation}
4. CityPopulation does not belong to any key.
5. CityPopulation is functionally dependent on the City which is a proper subset of the key

Example (Convert to 2NF)

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

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

New Scheme → {City, CityPopulation}

2NF - Decomposition

Example (Not 2NF)

Scheme → {<u>studio, movie</u>, budget, studio_city}

1. Key → {studio, movie}
2. {studio, movie} → {budget}
3. {studio} → {studio_city}
4. studio_city is not a part of a key
5. studio_city functionally depends on studio which is a proper subset of the key

Example (Convert to 2NF)

Old Scheme → {<u>Studio</u>, <u>Movie</u>, Budget, StudioCity}

New Scheme → {<u>Studio, Movie</u>, Budget}

New Scheme → {<u>Studio</u>, City}

Third Normal Form

- **Third normal form** (**3NF**) requires that data stored in a table be dependent only on the primary key, and not on any other field in the table.
- The database must meet all the requirements of the second normal form.
- Any field which is dependent not only on the primary key but also on another field is moved out to a separate table.

Third Normal Form (3NF)

This form dictates that all **non-key** attributes of a table must be functionally dependent on a candidate key i.e. there can be no interdependencies among non-key attributes.

No attribute is **transitively dependent** on the primary key

Example (Not in 3NF)

Scheme → {Title, PubID, PageCount, Price }

1. Key → {Title, PubId}
2. {Title, PubId} → {PageCount}
3. {PageCount} → {Price}
4. Both Price and PageCount depend on a key hence 2NF
5. Transitively {Title, PubID} → {Price} hence not in 3NF

3NF - Decomposition

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

Example (Convert to 3NF)

Old Scheme → {Title, PubID, PageCount, Price }

New Scheme → {PubID, PageCount, Price}

New Scheme → {Title, PubID, PageCount}

Third Normal Form (3NF)

Example (Not in 3NF)

Scheme → {Studio, StudioCity, CityTemp}

1. Primary Key → {Studio}
2. {Studio} → {StudioCity}
3. {StudioCity} → {CityTemp}
4. {Studio} → {CityTemp}
5. Both StudioCity and CityTemp depend on the entire key hence 2NF
6. CityTemp transitively depends on Studio hence violates 3NF

Example (Convert to 3NF)

Old Scheme → {Studio, StudioCity, CityTemp}

New Scheme → {Studio, StudioCity}

New Scheme → {StudioCity, CityTemp}

BuildingID	Contractor	Fee
100	Randolph	1200
150	Ingersoll	1100
200	Randolph	1200
250	Pitkin	1100
300	Randolph	1200

3NF - Decomposition

Example (Convert to 3NF)

Old Scheme → {BuildingID, Contractor, Fee}

New Scheme → {BuildingID, Contractor}

New Scheme → {Contractor, Fee}

Example (Not in 3NF)

Scheme → {BuildingID, Contractor, Fee}

1. Primary Key → {BuildingID}
2. {BuildingID} → {Contractor}
3. {Contractor} → {Fee}
4. {BuildingID} → {Fee}
5. Fee transitively depends on the BuildingID
6. Both Contractor and Fee depend on the entire key hence 2NF

BuildingID	Contractor
100	Randolph
150	Ingersoll
200	Randolph
250	Pitkin
300	Randolph

Contractor	Fee
Randolph	1200
Ingersoll	1100
Pitkin	1100

Boyce–Codd normal form (BCNF)

- **A relation is in BCNF if and only if every determinant is a candidate key.**
- Difference between 3NF and BCNF is that for a functional dependency A $\rightarrow$ B, 3NF allows this dependency in a relation if B is a primary-key attribute and A is not a candidate key.
- Whereas, BCNF insists that for this dependency to remain in a relation, A must be a candidate key.
- Every relation in BCNF is also in 3NF. However, relation in 3NF may not be in BCNF.

Boyce-Codd Normal Form (BCNF)

- Most 3NF relations are also BCNF relations.
- A 3NF relation is NOT in BCNF if:
  - Candidate keys in the relation are composite keys (they are not single attributes)
  - There is more than one candidate key in the relation, and
  - The keys are not disjoint, that is, some attributes in the keys are common.

1. Place the two candidate primary keys in separate entities.
2. Place each of the remaining data items in one of the resulting entities according to its dependency on the primary key.

BCNF - Decomposition

1. Example - (Not in BCNF)

Scheme → {City, Street, ZipCode }

1. Key1 → {City, Street }
2. Key2 → {ZipCode, Street}
3. No non-key attribute hence 3NF
4. {City, Street} → {ZipCode}
5. {ZipCode} → {City}
6. Dependency between attributes belonging to a key

Example (Convert to BCNF)

Old Scheme → {City, Street, ZipCode }

New Scheme1 → {ZipCode, Street}

New Scheme2 → {City, Street}

- Loss of relation {ZipCode} → {City}

Alternate New Scheme1 → {ZipCode, Street }

Alternate New Scheme2 → {ZipCode, City}

Boyce Codd Normal Form (BCNF)

Example - Movie (Not in BCNF)

Scheme → {MovieTitle, MovieID, PersonName, Role, Payment }

1. Key1 → {MovieTitle, PersonName}
2. Key2 → {MovieID, PersonName}
3. Both role and payment functionally depend on both candidate keys thus 3NF
4. {MovieID} → {MovieTitle}
5. Dependency between MovieID & MovieTitle Violates BCNF

Example (Convert to BCNF)

Old Scheme → {MovieTitle, MovieID, PersonName, Role, Payment }

New Scheme → {MovieID, PersonName, Role, Payment}

New Scheme → {MovieTitle, PersonName}

- Loss of relation {MovieID} → {MovieTitle}

New Scheme → {MovieID, PersonName, Role, Payment}

New Scheme → {MovieID, MovieTitle}

- We got the {MovieID} → {MovieTitle} relationship back

BCNF - Decomposition

Example - Consulting (Not in BCNF)

Scheme → {Client, Problem, Consultant}

1. Key1 → {Client, Problem}
2. Key2 → {Client, Consultant}
3. No non-key attribute hence 3NF
4. {Client, Problem} → {Consultant}
5. {Client, Consultant} → {Problem}
6. Dependency between attributes belonging to keys violates BCNF

Example (Convert to BCNF)

Old Scheme → {Client, Problem, Consultant}

New Scheme → {Client, Consultant}

New Scheme → {Client, Problem}

Multivalued Dependency(MVD)

- Dependency between attributes (for example, A, B, and C) in a relation, such that for each value of A there is a set of values for B and a set of values for C. However, the set of values for B and C are independent of each other.

- MVD between attributes A, B, and C in a relation uses the following notation:

  A −>> B A −>> C

- MVD can be further defined as being trivial or nontrivial.
  - ✓ MVD A −>> B in relation R is defined as being trivial if (a) B is a subset of A *or* (b) $A \cup B = R$.
  - ✓ MVD is defined as being nontrivial if neither (a) nor (b) are satisfied.

    A trivial MVD does not specify a constraint on a relation, while a nontrivial MVD does specify a constraint.

Fourth Normal Form

- Defined as a relation that is in Boyce-Codd Normal Form and contains no nontrivial multi-valued dependencies.
- Fourth normal form eliminates independent many-to-one relationships between columns.
- To be in Fourth Normal Form,
  - a relation must first be in Boyce-Codd Normal Form.
  - a given relation may not contain more than one multi-valued attribute.

BranchStaffOwner

branchNo	sName	oName
B003	Ann Beech	Carol Farrel
B003	David Ford	Carol Farrel
B003	Ann Beech	Tina Murphy
B003	David Ford	Tina Murphy

BranchStaff

branchNo	sName
B003	Ann Beech
B003	David Ford

BranchOwner

branchNo	oName
B003	Carol Farrel
B003	Tina Murphy

Fourth Normal Form (4NF)

Example (Not in 4NF)

Scheme → {MovieName, ScreeningCity, Type)

Primary Key: {MovieName, ScreeningCity, Type)

1. All columns are a part of the only candidate key, hence BCNF
2. Many Movies can have the same Type
3. Many Cities can have the same movie
4. Violates 4NF

Movie	Screening City	Type
Hard Code	Los Angles	Comedy
Hard Code	New York	Drama
Bill Durham	Santa Cruz	Drama
Bill Durham	Durham	comedy
The Code Warrier	New York	Horror

4NF - Decomposition

1. Move the two multi-valued relations to separate tables
2. Identify a primary key for each of the new entity.

Example (Convert to 3NF)

Old Scheme → {MovieName, ScreeningCity, Type}

New Scheme → {MovieName, ScreeningCity}

New Scheme → {MovieName, Type}

Movie	Type
Hard Code	Comedy
Bill Durham	Drama
The Code Warrier	Horror

Movie	ScreeningCity
Hard Code	Los Angles
Hard Code	New York
Bill Durham	Santa Cruz
Bill Durham	Durham
The Code Warrier	New York

Fourth Normal Form (4NF)

Example (Not in 4NF)

Scheme → {Manager, Child, Employee}

1. Primary Key → {Manager, Child, Employee}
2. Each manager can have more than one child
3. Each manager can supervise more than one employee
4. 4NF Violated

Manager	Child	Employee
Jim	Beth	Alice
Mary	Bob	Jane
Mary	NULL	Adam

Example (Convert to 4NF)

Old Scheme → {Manager, Child, Employee}

New Scheme → {Manager, Child}

New Scheme → {Manager, Employee}

Manager	Child
Jim	Beth
Mary	Bob

Manager	Employee
Jim	Alice
Mary	Jane
Mary	Adam

4NF - Decomposition

Example (Not in 4NF)

Scheme → {Employee, Skill, ForeignLanguage}

1. Primary Key → {Employee, Skill, Language }
2. Each employee can speak multiple languages
3. Each employee can have multiple skills
4. Thus violates 4NF

Employee	Skill	Language
1234	Cooking	French
1234	Carpentry	German
1453	Carpentry	German
1453	Cooking	Spanish
2345	Cooking	Spanish

Example (Convert to 4NF)

Old Scheme → {Employee, Skill, ForeignLanguage}

New Scheme → {Employee, Skill}

New Scheme → {Employee, ForeignLanguage}

Employee	Skill
1234	Cooking
1453	Carpentry
1453	Cooking
2345	Cooking

Employee	Language
1234	French
1234	German
1453	Spanish
2345	Spanish

Fifth Normal Form (5NF)

- Fifth normal form is satisfied when all tables are broken into as many tables as possible in order to avoid redundancy. Once it is in fifth normal form it cannot be broken into smaller relations without changing the facts or the meaning.
- A relation is in 5NF if and only if it has No Join Dependency.

Loss less Join Dependency

- It is a property of decomposition which ensures that no spurious tuples are generated when relations are reunion (rejoined) through natural join operation. It describes a type of dependency for a relation R with subset of attributes of R denoted as A,B, – – – – Z. A relation satisfies join dependency if and only if every Legal value of R is equal to the join of its projection on A,B, – – – – Z.

Property No.	Item Description	Supplier no.
PG4	BED	S1
PG4	CHAIR	S2
PG5	BED	S2
PG4	BED	S2

Property No.	Item Description
PG4	BED
PG4	CHAIR
PG5	BED
PG4	BED

Property No.	Supplier no.
PG4	S1
PG4	S2
PG5	S2
PG4	S2

Summary

First normal form- 1NF	-All values of the columns are atomic.
Second normal form- 2NF	-Must in 1NF. -no partial dependency.
Third normal form- 3NF	-Already in 2NF. -no transitive dependency in non key and key
Boyce-Codd normal form- BCNF	-No key attributes dependencies
Forth normal form- 4NF	-No multivalued dependencies
Fifth normal form- 5NF	-no join dependency