School of Computing Science and Engineering

Program: BCA

Course Code:BCAC2102

Course Name: Database Management System

Lecture-21

Topic- BCNF

Faculty:-Dr. Satyajee Srivastava

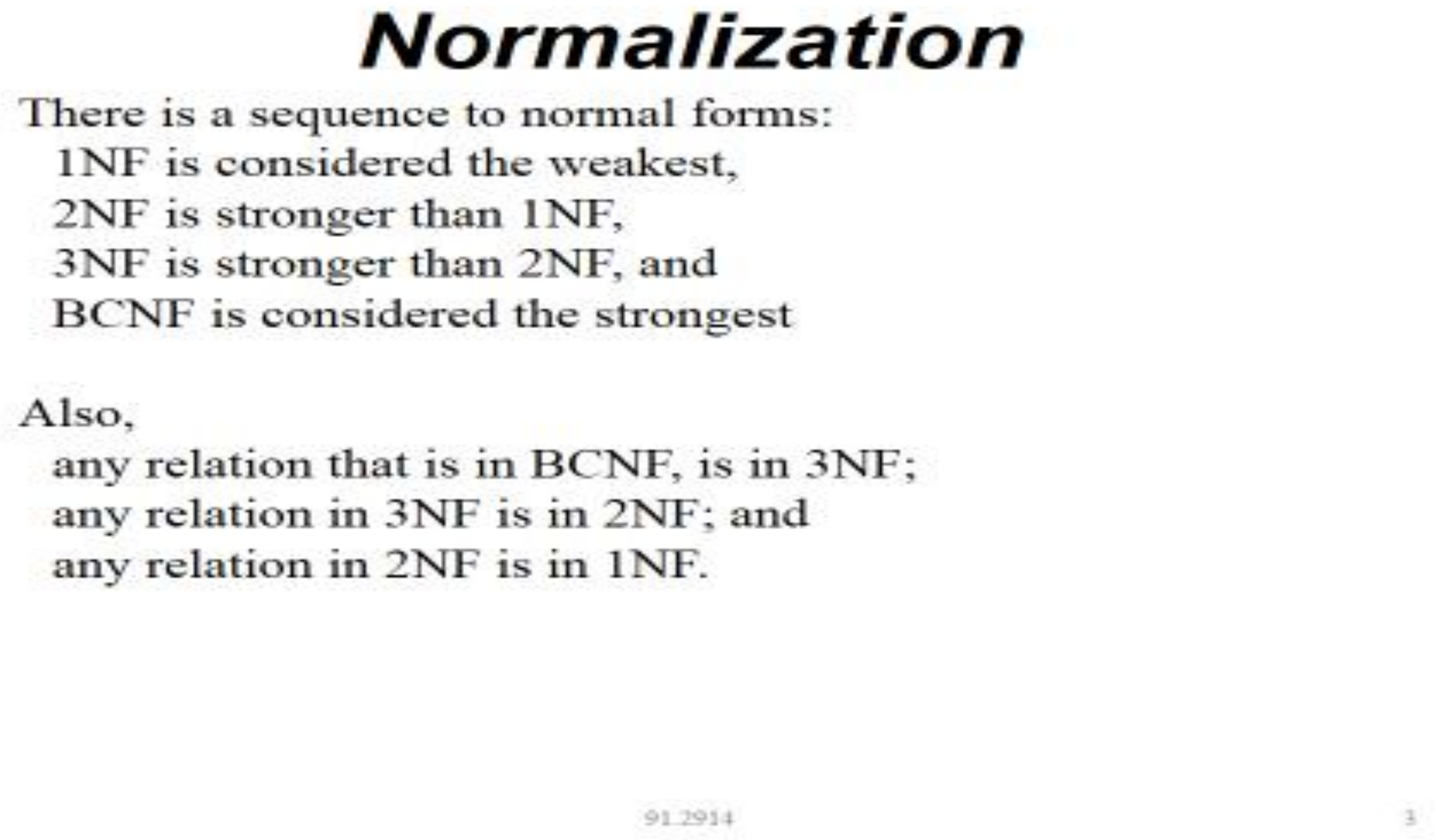
Lecture-20(RECAP)

Topic- 3NF

Objective :

To acquire knowledge about 3-Normal Forms

Lecture-20



Normalization

There is a sequence to normal forms:

1NF is considered the weakest,
2NF is stronger than 1NF,
3NF is stronger than 2NF, and
BCNF is considered the strongest

Also,

any relation that is in BCNF, is in 3NF;
any relation in 3NF is in 2NF; and
any relation in 2NF is in 1NF.

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 functionally dependent on the entire primary key

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

Example 1 (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

Second Normal Form (2NF)

CLASS EXERCISE

ARE FOLLOWING SCHEMAS ARE IN 2 NF.

1. Scheme → {City, Street, HouseNumber, HouseColor, CityPopulation}
2. Scheme → {studio, movie, budget, studio_city}

Second Normal Form (2NF)

Example 2 (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 3 (Not 2NF)

Scheme → {studio, movie, 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

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 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 → {Studio, Movie, Budget, StudioCity}

New Scheme → {Movie, Studio, Budget}

New Scheme → {Studio, City}

Example 3 (Convert to 2NF)

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

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

New Scheme → {City, CityPopulation}

Functional Dependencies

1. 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	Balloon	$34.00
0-55-123456-9	Main Street	$22.95
0-123-45678-0	Ulysses	$34.00
1-22-233700-0	Visual Basic	$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}

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.

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

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

Third Normal Form (3NF)

Example 2 (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 3 (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	Fee
100	Randolph	1200
150	Ingersoll	1100
200	Randolph	1200
250	Pitkin	1100
300	Randolph	1200

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 1 (Convert to 3NF)

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

New Scheme → {PubID, PageCount, Price}

New Scheme → {Title, PubID, PageCount}

3NF - Decomposition

Example 2 (Convert to 3NF)

Old Scheme → {Studio, StudioCity, CityTemp}

New Scheme → {Studio, StudioCity}

New Scheme → {StudioCity, CityTemp}

Example 3 (Convert to 3NF)

Old Scheme → {BuildingID, Contractor, Fee}

New Scheme → {BuildingID, Contractor}

New Scheme → {Contractor, Fee}

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

Contractor	Fee
Randolph	1200
Ingersoll	1100
Pitkin	1100

Lecture-20

CLASS -ASSIGNMENT

Explain about Full functional dependency and Partial dependency

Lecture-21

Topic- BCNF

Objective :

To acquire knowledge about 3-Normal Forms

Lecture-21

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

  (A determinant is any attribute whose value determines other values with a row.)

- **If a table contains only one candidate key, the 3NF and the BCNF are equivalent.**
- **BCNF is a special case of 3NF.**

Boyce-Codd Normal Form (BCNF)

- BCNF does not allow dependencies between attributes that belong to candidate keys.
- 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.
- Third normal form and BCNF are not same if the following conditions are true:
  - The table has two or more candidate keys
  - At least two of the candidate keys are composed of more than one attribute
  - The keys are not disjoint i.e. The composite candidate keys share some attributes

Example 1 - Address (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**

Boyce Codd Normal Form (BCNF)

Example 2 - 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 3 - 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 attributess belonging to keys violates BCNF

BCNF - Decomposition

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.

Example 1 (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}

Decomposition – Loss of Information

1. If decomposition does not cause any loss of information it is called a **lossless** decomposition.
2. If a decomposition does not cause any dependencies to be lost it is called a **dependency-preserving** decomposition.
3. Any table scheme can be decomposed in a lossless way into a collection of smaller schemas that are in BCNF form. However the dependency preservation is not guaranteed.
4. 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

Use your own judgment when decomposing schemas

BCNF - Decomposition

Example 2 (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

Example 3 (Convert to BCNF)

Old Scheme → {Client, Problem, Consultant}

New Scheme → {Client, Consultant}

New Scheme → {Client, Problem}

Thank You