DA 512H: Database Management Systems

Schema Refinement (2)

Debanga Raj Neog
Mehta Family School of Data Science and Artificial Intelligence (MFSDS&AI)
IIT Guwahati

Lecture Plan

- Normal Forms
  - 1NF, 2NF
  - “Good” Vs “Bad” FDs
  - BCNF and 3NF
- BCNF Decomposition algorithm
- Examples

Normal Forms

Does the schema require refinement?

- If a relation is in a certain **normal form** (BCNF, 3NF etc.), it is known that certain kinds of problems are avoided/minimized.
- This can be used to help us decide whether *decomposing* the relation will help.

Normal Forms

Normal Form types:

- First normal form (1NF)
- Second normal form (2NF)
- Third normal form (3NF)
- Boyce-Codd normal form (BCNF)

The normal forms are **increasingly restrictive** from 1NF to BCNF

- Every relation in BCNF is also in 3NF, every relation in 3NF is also in 2NF, and every relation in 2NF is in 1NF.

1st Normal Form (1NF)

- A relation is in first normal form (**1NF**) if every field contains only atomic values, that is, not lists or sets.
- This requirement is implicit in our definition of the relational model.

Student	Course
Mahesh	{MA518, CS348}
Paes	{MA518, MA251}

Violates 1NF

Student	Course
Mahesh	MA518
Mahesh	CS348
Paes	MA518
Paes	MA251

In 1NF

1NF Constraint: Types must be atomic!

Second Normal Form (2NF)

- For a table to be in 2NF, there are two requirements
  - The database is in first normal form
  - All non-key attributes in the table must be functionally dependent on the entire primary key (does not have partial dependencies)

Second Normal Form (2NF)

- For a table to be in 2NF, there are two requirements
  - The database is in first normal form
  - All non-key attributes in the table must be functionally dependent on the entire primary key (does not have partial dependencies)
- **Example: Is it 2NF?**

Schema: R {Title, PubId, AuId, Price, AuAddress}

FDs:

$\{Title, PubId, AuID\} \longrightarrow \{Price\}$

$\{AuID\} \longrightarrow \{AuAddress\}$

PK — Non-key

Title	PubId	AuId	Price	AuAddress

Quick Revision: Keys

- **Relation**

Emp_SSN	Emp_Id	Emp_name	Emp_email
11051	01	John	john@email.com
19801	02	Merry	merry@email.com
19801	03	Riddle	riddle@email.com
41201	04	Cary	cary@email.com

- **Super Keys:**

Set of super keys obtained
{ Emp_SSN }
{ Emp_Id }
{ Emp_email }
{ Emp_SSN, Emp_Id }
{ Emp_Id, Emp_name }
{ Emp_SSN, Emp_Id, Emp_email }
{ Emp_SSN, Emp_name, Emp_Id }

- Superset of Candidate Keys
- May contain redundant attribute

- **Candidate Keys:**

Candidate Keys :
Emp_SSN
Emp_Id
Emp_email

- Should not contain redundant attribute
- Should be minimal, no proper subset should be a candidate key

- **Primary Key:** We can pick the most appropriate attribute as a Primary Key

Third Normal Form (3NF)

Let R be a relation schema, X be a subset of the attributes of R, and A be an attribute of R.

R is in **third normal form (3NF)** if for every FD $X \rightarrow A$ that holds over R, one of the following statements is true:

- $A \in X$; that is, it is a trivial FD, or
- X is a superkey, or
- A is part of some (candidate) key for R.

Third Normal Form (3NF)

Let R be a relation schema, X be a subset of the attributes of R, and A be an attribute of R.

R is in **third normal form (3NF)** if for every FD $X \rightarrow A$ that holds over R, one of the following statements is true:

- $A \in X$; that is, it is a trivial FD, or
- X is a superkey, or
- A is part of some (candidate) key for R.

Note: Finding all keys of a relation schema is known to be an NP-complete problem, and so is the problem of determining whether a relation schema is in 3NF.

Question:

- How will you check that X is a superkey?

Problem: Given

- R(X Y Z W)
- FD= { XYZ $\longrightarrow$ W, XY $\longrightarrow$ ZW and X $\longrightarrow$ YZW}

Is XYZ a super key?

Question:

- Attribute closure should contain all attributes of R

Question:

Problem: Given

- A relation R(X, Y, Z, W, P) and
- Functional Dependency set FD = { $X \longrightarrow Y$, $Y \longrightarrow P$, and $Z \longrightarrow W$}
- Candidate Key is XZ

Determine whether the given R is in 3NF.

Boyce-Codd Normal Form (BCNF)

Let R be a relation schema, X be a subset of the attributes of R, and A be an attribute of R.

R is in **Boyce-Codd Normal Form (BCNF)** if for every FD $X \rightarrow A$ that holds over R, one of the following statements is true:

- $A \in X$; that is, it is a trivial FD, or
- X is a superkey

BCNF versus 3NF

- If R is in BCNF, it's obviously in 3NF.
- If R is in 3NF, some redundancy is possible.
- Thus, 3NF is indeed a compromise relative to BCNF when BCNF not achievable
- *Lossless-join, dependency-preserving decomposition of R into a collection of 3NF relations is always possible*

Ex: 3NF

- **Reserves**(sid, bid, day, card_no)
- Key: {sid, bid, day}
- Every sailor has unique credit card, sid → card_no
  - Partial dependency, thus NOT in 3NF
- But we have the FD, card_no → sid
  - Therefore candidate key: card_no,bid, day
  - Now card_no is part of a key, so it is in 3NF

Ex: BCNF

- Is this table in BCNF?

Name	SIN	PhoneNumber	City
Fred	123-45-6789	604-555-1234	Vancouver
Fred	123-45-6789	604-555-6543	Vancouver
Joe	987-65-4321	908-555-2121	Burnaby
Joe	987-65-4321	908-555-1234	Burnaby

$\{SIN\} \rightarrow \{Name, City\}$

Ex: BCNF

- Is this table in BCNF?

Name	SIN	PhoneNumber	City
Fred	123-45-6789	604-555-1234	Vancouver
Fred	123-45-6789	604-555-6543	Vancouver
Joe	987-65-4321	908-555-2121	Burnaby
Joe	987-65-4321	908-555-1234	Burnaby

$\{SIN\} \rightarrow \{Name, City\}$

$\Longrightarrow$ <u>Not</u> in BCNF

Ex: BCNF

- How can we make it BCNF?

Name	SIN	PhoneNumber	City
Fred	123-45-6789	604-555-1234	Vancouver
Fred	123-45-6789	604-555-6543	Vancouver
Joe	987-65-4321	908-555-2121	Burnaby
Joe	987-65-4321	908-555-1234	Burnaby

$\{SIN\} \rightarrow \{Name, City\}$

$\Longrightarrow$ Not in BCNF

Ex: BCNF

Name	SIN	City
Fred	123-45-6789	Vancouver
Joe	987-65-4321	Burnaby

SIN	PhoneNumber
123-45-6789	604-555-1234
123-45-6789	604-555-6543
987-65-4321	908-555-2121
987-65-4321	908-555-1234

$\{SIN\} \longrightarrow \{Name,City\}$

This FD is now *good* because it is the key

Now in BCNF!

Is there some algorithm to convert to convert a relation scheme to BCNF?

BCNF Decomposition Algorithm

BCNFDecomp(R):

BCNF Decomposition Algorithm

BCNFDecomp(R):
  Find *a non-trivial bad FD:* $X \longrightarrow Y$

X is not a key, i.e.,
$X^{+} \neq$ [all attributes]

BCNF Decomposition Algorithm

BCNFDecomp(R):
  Find *a non-trivial bad FD:* $X \rightarrow Y$

  if (not found) **then** **Return** R

If no "bad" FDs found, in BCNF!

BCNF Decomposition Algorithm

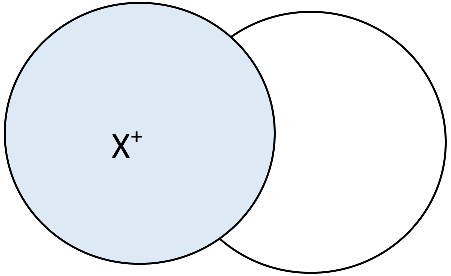
BCNFDecomp(R):

Find *a non-trivial bad FD: X ⟶ Y*

if (not found) **then** **Return** R

Split R into X^+ and X ∪[remaining attributes]

One table is X^+



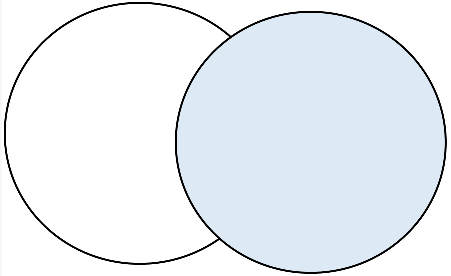
BCNF Decomposition Algorithm

BCNFDecomp(R):
  Find *a non-trivial bad FD: X ⟶ Y*

  <u>if</u> (not found) **<u>then</u> Return** R

  Split R into X^+ and X ∪[remaining attributes]

The other table is
X ∪ (R – X^+)



BCNF Decomposition Algorithm

BCNFDecomp(R):
  Find *a non-trivial bad FD: X Y*

  if (not found) **then** **Return** R

  Split R into X^+ and X ∪ [rest attributes]

  Return BCNFDecomp(R_1), BCNFDecomp(R_2)

Proceed recursively until no more “bad” FDs!

Example

Student	Course	Room
Mahesh	MA518	C1-101
Paes	MA518	C1-101
Sindhu	MA518	C1-101
..	..	..

Course $\rightarrow$ Room

$X \cup (R - X^{+})$

X^{+}

Student	Course
Mahesh	MA518
Paes	MA518
Sindhu	MA518
..	..

Course	Room
MA518	C1-101
CS348	C2-101

Exercise - 2

BCNFDecomp(R):

Find *a non-trivial bad FD:* $X \rightarrow Y$

if (not found) **then Return** R

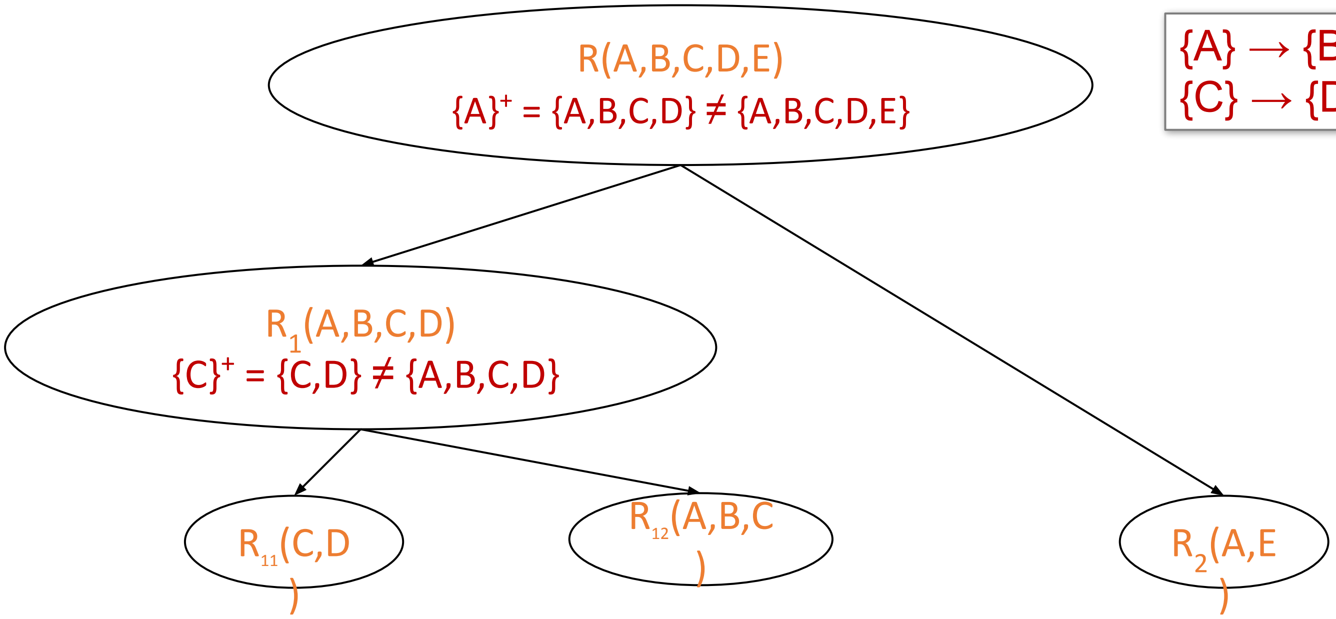
Split R into X^+ and $X \cup$[rest attributes]

Return BCNFDecomp(R_1), BCNFDecomp(R_2)

R(A,B,C,D,E)

$\{A\} \rightarrow \{B,C\}$
$\{C\} \rightarrow \{D\}$

Exercise - 2



R(A,B,C,D,E)

$\{A\} \rightarrow \{B,C\}$
$\{C\} \rightarrow \{D\}$

3NF to BCNF

Problem:

- R(A,B,C,D)
- F = {AB $\longrightarrow$ CD, C $\longrightarrow$A, D$\longrightarrow$B}