Normalization Algorithm

1. Identify functional dependencies (try to involve as many attributes as possible)

2. Find candidate keys by applying the inference rules

X is a candidate key iff X $\rightarrow$ A1,...,An \ X and X is minimal

(in large relational schema there are usually more than one)

3. Find and mark all **prime (X)** and **non-prime** attributes

4. Choose one of the candidate keys for a primary key

(5) 1NF (your relation is already in 1NF if you have followed the translation algorithm)

Normalization Algorithm

6. 2NF: (*Make sure your tables are in 1NF.*)
Question: Are there **non-prime** attributes functionally dependent on **a part of a candidate key**?
Yes: Split the tables by moving the determining and determined attributes to a new table. Remove the determined attributes from the old table and restart the algorithm for both tables.
No: Continue to 3NF

7. 3NF: *Make sure your tables are in 2NF.*
Question: Are there **non-prime** attributes functionally dependent on something that is **not a candidate key**?
Yes: Split the tables by moving the determining and determined attributes to a new table. Remove the determined attributes from the old table and restart the algorithm for both tables.
No: Continue to BCNF

8. BCNF: *Make sure your tables are in 3NF.*
Question: Does it exist a functional dependency for which the **determinant** is **not a candidate key?**
Yes: Split the tables by moving the determining and determined attributes to a new table. Remove the determined attributes from the old table and restart the algorithm for both tables.

No: Done

Normalization

Personal Number	Student Name	StudentID	Course Code	Course Name	Exam Moments	Examiner	Email
19890723-1324	Harry Potter	harpo581	course1	dark arts	{exam, practical exercise}	P. McGonagall	pmc@hogwarts.co.uk
19890723-1324	Harry Potter	harpo581	course2	transformation	{laboration, home exam}	P. McGonagall	pmc@hogwarts.co.uk
19890723-1324	Harry Potter	harpo581	course3	potions	{laboration}	S. Snape	ssn@hogwarts.co.uk
19880824-3422	Ron Weasley	rowea982	course1	dark arts	{exam, practical exercise}	P. McGonagall	pmc@hogwarts.co.uk
19880824-3422	Ron Weasley	rowea982	course2	transformation	{laboration, home exam}	P. McGonagall	pmc@hogwarts.co.uk
19880824-3422	Ron Weasley	rowea982	course3	potions	{laboration}	S. Snape	ssn@hogwarts.co.uk
19870922-2135	Draco Malfoy	drama001	course1	dark arts	{exam, practical exercise}	P. McGonagall	pmc@hogwarts.co.uk
19870922-2135	Draco Malfoy	drama001	course3	potions	{laboration}	S. Snape	ssn@hogwarts.co.uk

Step 1, Functional dependencies:

StudentID->Personal number, StudentName

Course Code->Course Name, Exam Moments, Examiner

Examiner->Email

Personal Number -> Student ID

Assumptions:

Student names not unique

Course names not unique

One email per examiner

Examiner is unique

Only one examiner per course

Step 2, Candidate keys:

(1) Course Code → Course Name, Exam Moments, Examiner
imply **Course Code → Examiner** (decomposition)

(2) Course Code → Examiner and Examiner → Email
imply **Course Code → Email** (transitive)

(3) Course Code → Course Name, Exam Moments, Examiner
and Course Code → Email
imply **Course Code → Course Name, Exam Moments, Examiner, Email** (union)

(4) Course Code → Course Name, Exam Moments, Examiner, Email
imply **Course Code, StudentID → StudentID, Course Name, Exam Moments, Examiner, Email** (augmentation)

(5) Course Code, StudentID → StudentID, Course Name, Exam Moments, Examiner, Email
imply **Course Code, StudentID → Course Name, Exam Moments, Examiner, Email** (decomposition)

(6) StudentID → Personal number, StudentName imply **Course Code, StudentID → Course Code, Personal number, StudentName** (augmentation)

(7) Course Code, StudentID → Course Code, Personal number, StudentName imply **Course Code, StudentID → Personal number, StudentName** (decomposition)

(8) (5) and (7) imply **Course Code, StudentID → Personal number, StudentName, Course Name, Exam Moments, Examiner, Email** (union)

i.e. **(StudentID, Course Code)** is a candidate key

Similarly **(Personal Number, Course Code)** is also a candidate key

Step 3, Prime attributes: Personal Number, StudentId, Course Code

Non-prime attributes: Student Name, Course Name, Exam Moments, Examiner, Email

Step 4: We choose **(Personal Number, Course Code)** for primary key;

Step 5: 1NF

Prime attributes: Personal Number, StudentId, Course Code

Non-prime attributes: Student Name, Course Name, Exam Moments, Examiner, Email

- 1NF: Split all non-atomic values
- Before:

Personal number	Student Name	StudentID	Course Code	Course Name	Exam Moments	Examiner	Email

- After:

Course Code	Exam Moments

Personal number	Student Name	StudentID	Course Code	Course Name	Examiner	Email

Prime attributes: Personal Number, StudentId, Course Code

Non-prime attributes: Student Name, Course Name, Exam Moments, Examiner, Email

Step 6: 2NF

- 2NF: No nonprime-attribute should be dependent on part of candidate key
- Before:

Course Code	Exam Moments

Personal number	Student name	StudentID	Course Code	Course Name	Examiner	Email

- After:

Course Code	Exam Moments

Personal number	Student name

Course Code	Course Name	Examiner	Email

Personal number	StudentID	Course Code

Prime attributes: Personal Number, StudentId, Course Code

Non-prime attributes: Student Name, Course Name, Exam Moments, Examiner, Email

Step 7: 3NF

- 3NF: No non-prime attribute should be dependent on any other set of attributes which is not a candidate key
- Before:

Course Code	Exam Moments

Personal number	Student name

Course Code	Course Name	Examiner	Email

Personal number	StudentID	Course Code

- After:

Course Code	Exam Moments

Personal number	Student Name

Course Code	Course Name	Examiner

Examiner	Email

Personal number	StudentID	Course Code

Step 8: BCNF

Prime attributes: Personal Number, StudentId, Course Code

Non-prime attributes: Student Name, Course Name, Exam Moments, Examiner, Email

- BCNF: Every determinant is a candidate key
- Before:

Course Code	Exam Moments	
Personal number	Student name	
Course Code	Course Name	Examiner
Examiner	Email	
Personal number	StudentID	Course Code

- After:

Course Code	Exam Moments	
Personal number	Student Name	
Course Code	Course Name	Examiner
Examiner	Email	
Personal number	Course Code	
Personal number	StudentID	

Example 0

Given the relation **R(A, B, C, D, E, F)** with functional dependencies {**A** $\rightarrow$ **BC, C** $\rightarrow$ **AD,**
DE $\rightarrow$ **F**},

1. Find all the candidate keys of R. Use the inference rules in the course to reach your conclusion. Do not use more than one rule in each derivation step.

2. Normalize R to BCNF. Explain the process step by step.

Step 1: The functional dependencies are given;

Example 0 - Solution

Step 2: We now show that AE is a candidate key.

(1) A → **BC** implies **A** → **C** (decomposition)

(2) C → **DA** implies **C** → **D** (decomposition)

(3) A → **C** and **C** → **D** imply **A** → **D** (transitive rule (1) and (2))

(4) A → **D** implies **AE** → **DE** (augmentation)

(5) AE → **DE** and **DE** → **F** implies **AE** → **F** (transitive rule (4) and (DE → F))

(6) A → **BC** and **A** → **D** imply **A** → **BCD** (union (A → BC) and (3))

(7) A → **BCD** implies **AE** → **BCDE** (augmentation with E)

(8) AE → **BCDE** implies **AE** → **BCD** (decomposition)

(9) AE → **BCD** and **AE** → **F** implies **AE** → **BCDF** (union (8) and (5))

Example 0 - Solution

We now show that CE is a also candidate key.

(10) C → **DA** implies **C** → **A** (decomposition)

(11) C → **A** implies **CE** → **AE** (augmentation with E)

(12) CE → **AE** and **AE** → **BCDF** implies **CE** → **BCDF**

(transitive rule (11) and (9))

(13) CE → **BCDF** implies **CE** → **BDF** (decomposition)

(14) CE → **AE** implies **CE** → **A** (decomposition)

(15) CE → **A** and **CE** → **BCDF** imply **CE** → **ABDF** (union (14) and (13))

Step 3: Prime attributes: A, C, E

Non-prime attributes B, D, F

Example 0 - Solution

Step 5: Already in 1NF since there are no non-atomic values

Step 6: Since **A → BD** violates the definition of 2NF, we have to split the original table into: *(from (6) A → BCD, however C is prime, i.e., we may or may not move it with B and D)*

R1(A,C,E,F) with AE and CE as candidate keys and functional dependencies {**AE → F, A → C, CE → F, C → A** }

R2(A, B, D) with A as candidate key and functional dependencies{**A → BD**}

Now, R1 and R2 satisfy the definition of 2NF.

Example 0 - Solution

Step 7: Relations R1 and R2 are already in 3NF since there are no non-prime attributes which are dependent on a set of attributes that is not a candidate key.

Step 8: Relation R2 is in BCNF since every determinant (A in this case) is a candidate key.

Relation R1 is not in BCNF since determinant C (or A) is not a candidate key. Therefore, we need to split R1 into:

R11(A, E, F) with AE as candidate key and functional dependencies {**AE** → **F**} and

R12(A, C) with A and C as candidate keys and functional dependencies{**A** → **C**, **C** → **A**}

Example 1

Given the relation **R(A, B, C, D, E, F, G, H)** with functional dependencies {**AB** → **CDEFGH**, **CD** → **B**, **D** → **EFGH**, **E** → **FGH**, **FG** → **E**, **G** → **H**},

1. Find all the candidate keys of R. Use the inference rules in the course to reach your conclusion. Do not use more than one rule in each derivation step.

2. Normalize R to 2NF. Explain the process step by step.

Step 1: The functional dependencies are given;

LiU EXPANDING REALITY

Example 1 - Solution

Step 2: The functional dependency **AB → CDEFGH** implies that AB is a candidate key. We now show that ACD is also a candidate key.

AB → CDEFGH implies **AB → EFGH** (decomposition)

AB → EFGH and **CD → B** imply **ACD → EFGH** (pseudotransitive)

CD → B implies **ACD → AB** (augmentation)

ACD → AB implies **ACD → B** (decomposition)

ACD → B and **ACD → EFGH** imply **ACD → BEFGH** (union)

Example 1 - Solution

Step 3: The solution to Step 2 implies that A, B, C and D are prime and E, F, G and H non-prime.

Step 6: Since **D → EFGH** violates the definition of 2NF, we have to split the original table into

R(A,B,C,D) with AB and ACD as candidate keys and functional dependencies {**AB → CD, CD → B**}

R2(D,E,F,G,H) with D as candidate key and functional dependencies{**D → EFGH, E → FGH, FG → E, G → H**}

Now, R and R2 satisfy the definition of 2NF.

Example 2

Normalize (1NF → 2NF → 3NF → BCNF) the relation **R(A, B, C, D, E, F, G, H)** with functional dependencies F={**ABC** → **DEFGH, D** → **CEF, EF** → **GH**}. Explain your solution step by step.

Step 1: The functional dependencies are given;

Example 2 - Solution

Step 2: The functional dependency **ABC → DEFGH** implies that ABC is a candidate key. We now show that ABD is also a candidate key.

ABC → DEFGH implies **ABC → EFGH** (decomposition)

D → CEF implies **D → C** (decomposition)

ABC → EFGH and **D → C** imply **ABD → EFGH** (pseudotransitive)

D → C implies **ABD → C** (augmentation)

ABD → C and **ABD → EFGH** imply **ABD → CEFGH** (union)

Example 2 - Solution

Step 3: The candidate keys above imply that A, B, C and D are prime and E, F, G and H non-prime.

Step 6: Since **D** → **EFGH** violates the definition of 2NF, we have to split the original table into

R1(A,B,C,D) with ABC and ABD as candidate keys and functional dependencies {**ABC** → **D**, **D** → **C**}

R2(D,E,F,G,H) with D as candidate key and functional dependencies {**D** → **EFGH**, **EF** → **GH**}.

Example 2 - Solution

Step 7: Now, R1 and R2 satisfy the definition of 2NF. However, R2 does not satisfy the definition of 3NF due to **EF** → **GH**. Then, we have to split R2 into

R21(D,E,F) with D as candidate key and functional dependencies {**D** → **EF**}

R22(E,F,G,H) with EF as candidate key and functional dependencies {**EF** → **GH**}.

Example 2 - Solution

Step 8: Now, R1, R21 and R22 satisfy the definition of 3NF. However, R1 does not satisfy the definition of BCNF due to **D** → **C**. Then, we have to split R1 into

R11(A,B,D) with candidate key A,B,D.

R12(D,C) with candidate key D

CarSale Example

Consider the following relation **CarSale(Car#, Salesman#, Commission, DateSold, Discount)**. Assume that a car may be sold by multiple salesman and hence Car#,Salesman# is the primary key. Additional dependencies are:

DateSold → Discount

Salesman# → Commission

Based on the given primary key, is the relation in 1NF, 2NF or 3NF? Why or why not? How would you successively normalize it completely?

CarSale Example - Solution

Car#,Salesman# is the primary key, that is:

Car#,Salesman# → Commission, DateSold, Discount

Step 3: Prime attributes Car#,Salesman#, the rest non-prime;

Step 5: It is in 1NF

Step 6: Not 2NF because of Salesman# → Commission

R1 (Car#, Salesman#, DateSold, Discount)

R2 (Salesman#, Commission)

CarSale Example

Step 7:

R1 Not in 3NF:

R1 (Car#, Salesman#, DateSold, Discount)

is not in 3NF because of **DateSold** → **Discount**

R11 (Car#, Salesman#, DateSold)

R12 (DateSold, Discount)