



UNIVERSITAS
INDONESIA
Visitas, Profektas, Inovitas

FAKULTAS
ILMU
KOMPUTER

Slide 12

Normalisasi

CSF2600700 - BASIS DATA
SEMESTER GENAP 2017/2018



Tujuan Pemelajaran

Setelah mengikuti pemelajaran pada topik ini, Anda diharapkan dapat merancang skema basis data yang baik, mencapai bentuk normal 3NF atau BCNF.



Outline

1. Panduan Informal dalam Merancang Basis Data Relasional
2. *Functional Dependency*
3. Normalisasi Berdasarkan *Primary Key*
4. *General Normal Form*
5. *Boyce Codd Normal Form*



Panduan Informal dalam Merancang Basis Data Relasional



Panduan Merancang Basis Data Relasional

- Skema basis data dapat dilihat dari dua level:
 - pengguna (logical level)
 - penyimpanan data di komputer (*storage level*)
- Perancangan di sini berfokus pada skema basis data yang tersimpan di komputer (*base relations*)
- Bagaimana kriteria *base relations* yang baik?



Panduan Informal

Panduan secara umum berkaitan dengan:

- Semantik atribut-atribut relasi
- Data yang berulang dan kaitannya dengan *update anomaly*
- Nilai null
- *Spurious tuples*

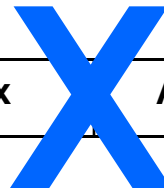


Panduan 1: Merepresentasikan Satu Entity

Panduan 1: Setiap tuple pada relasi seharusnya merepresentasikan satu entity atau relationship instance. Skema perlu dirancang agar mudah dijelaskan relasi demi relasi. Semantik dari atribut sebaiknya mudah diinterpretasikan

Atribut-atribut dari entity yang berbeda-beda seharusnya tidak bergabung dalam satu relasi
Hanya foreign key yang digunakan untuk mengacu ke relasi lain
Atribut-atribut entity dan relationship sebaiknya dipisahkan

SSN	Name	BirthData	Sex	Address	DNo	DName	DMgrSSN
-----	------	-----------	-----	---------	-----	-------	---------



SSN	Name	BirthData	Sex	Address	DNo
-----	------	-----------	-----	---------	-----

DNo	DName	DMgrSSN
-----	-------	---------



Panduan 2: Minimalisir Nilai *NULL*

- Panduan 2: *Relations* harus dirancang agar nilai *null* yang ada pada baris-barisnya sesedikit mungkin
- *Attributes* yang bernilai *null* seringkali dapat ditempatkan pada *relations* terpisah.



Panduan 3: *Lossless join condition*

- Dalam merancang, kadang-kadang kita melakukan dekomposisi relasi maupun penggabungan beberapa relasi. Ada 2 hal yang perlu diperhatikan:
 - Apakah dekomposisi (pemecahan) sebuah relasi menjadi dua/lebih relasi baru akan menyebabkan **adanya data yang hilang**?
 - Apakah penggabungan dua/lebih relasi menjadi sebuah relation baru akan menghasilkan kelebihan data (yang sebenarnya tidak ada)?

Panduan 3: *Lossless join condition*

○ Panduan 3:

- Relasi harus dirancang agar memenuhi *lossless join condition* dan
- tidak ada *spurious tuples* yang dihasilkan oleh *natural join* antar relasi

Panduan 4: Hindari Anomali

A poorly designed database causes *anomalies*:

Student	Course	Room
Mary	CS145	B01
Joe	CS145	B01
Sam	CS145	B01
..

If every course is in only one room, contains redundant information!

Panduan 4: Hindari Anomali

A poorly designed database causes *anomalies*:

Student	Course	Room
Mary	CS145	B01
Joe	CS145	C12
Sam	CS145	B01
..

If we update the room number for one tuple, we get inconsistent data = an update anomaly

Panduan 4: Hindari Anomali

A poorly designed database causes *anomalies*:

Student	Course	Room
..

If everyone drops the class, we lose what room the class is in! = a *delete anomaly*

Panduan 4: Hindari Anomali

A poorly designed database causes *anomalies*:

Student	Course	Room
Mary	CS145	B01
Joe	CS145	B01
Sam	CS145	B01
..

Similarly, we can't reserve a room without students = an insert anomaly



...	CS229	C12
-----	-------	-----

Is this form better?

Student	Course
Mary	CS145
Joe	CS145
Sam	CS145
..	..

Course	Room
CS145	B01
CS229	C12

Today: develop theory to understand why this design may be better **and** how to find this *decomposition*...



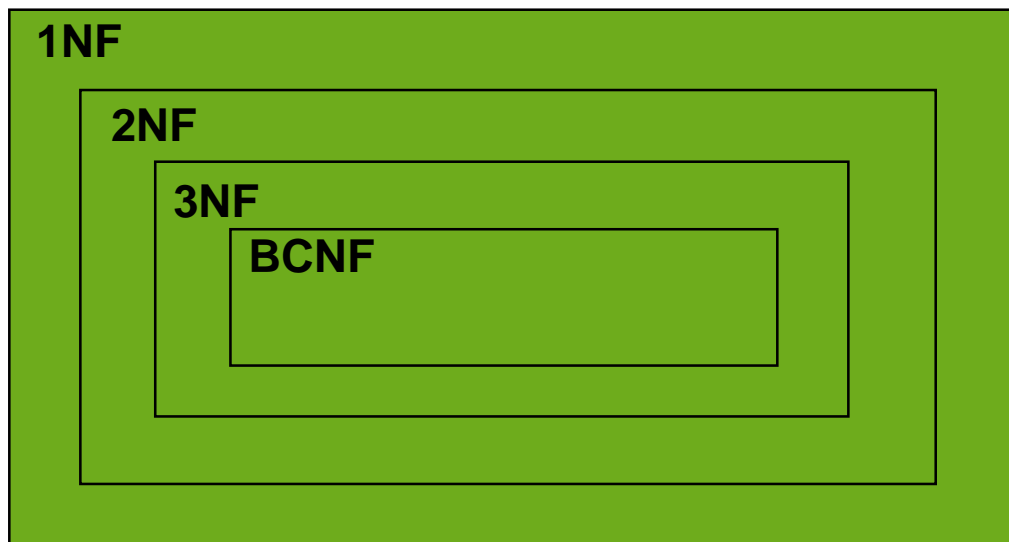
Normalisasi

- Normalisasi (Normalization):
 - Proses dekomposisi relasi yang masih “buruk” dengan memecah atribut-atributnya untuk membentuk beberapa relasi
- Bentuk Normal (Normal Form)
 - Kondisi (dengan menggunakan FD dan key) yang menentukan apakah suatu skema relasi memenuhi kriteria tertentu



Bentuk Normal

- 1st Normal Form (1NF) = All tables are flat
 - 2nd Normal Form
 - 3rd Normal Form (3NF)
 - Boyce-Codd Normal Form (BCNF)
 - 4th and 5th Normal Forms = *see text books*
- DB designs based on *functional dependencies*, intended to prevent data ***anomalies***



Functional Dependency



Definisi

Def: Let A, B be sets of attributes

We write $A \rightarrow B$ or say A *functionally determines* B if, for any tuples t_1 and t_2 :

$$t_1[A] = t_2[A] \text{ implies } t_1[B] = t_2[B]$$

and we call $A \rightarrow B$ a functional dependency

$A \rightarrow B$ means that

“whenever two tuples agree on A then they agree on B .”

A Picture Of FDs

	A ₁	...	A _m		B ₁	...	B _n	

Defn (again):

Given attribute sets $A=\{A_1,\dots,A_m\}$ and $B = \{B_1,\dots,B_n\}$ in R ,

A Picture Of FDs



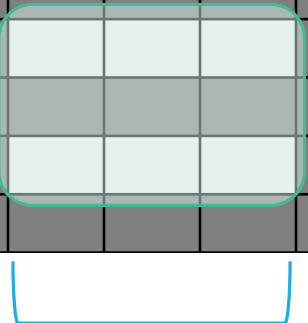
Defn (again):

Given attribute sets $A=\{A_1,\dots,A_m\}$ and $B = \{B_1,\dots,B_n\}$ in R ,

The *functional dependency* $A \rightarrow B$ on R holds if for *any* t_i, t_j in R :

A Picture Of FDs

	A_1	...	A_m		B_1	...	B_n	
t_i								
t_j								



If t_1, t_2 agree
here..

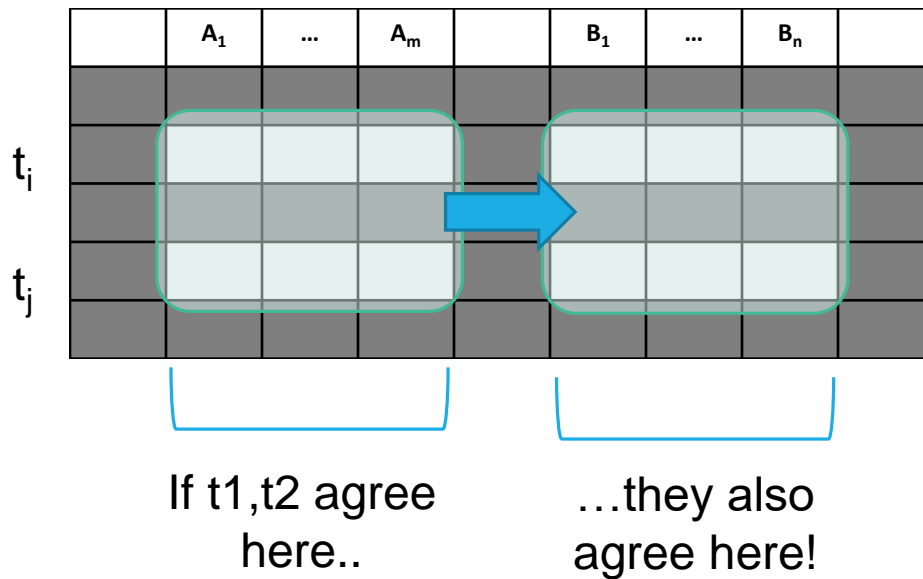
Defn (again):

Given attribute sets $A = \{A_1, \dots, A_m\}$ and $B = \{B_1, \dots, B_n\}$ in R ,

The *functional dependency* $A \rightarrow B$ on R holds if for *any* t_i, t_j in R :

$t_i[A_1] = t_j[A_1]$ AND $t_i[A_2] = t_j[A_2]$ AND
... AND $t_i[A_m] = t_j[A_m]$

A Picture Of FDs



Defn (again):

Given attribute sets $A = \{A_1, \dots, A_m\}$ and $B = \{B_1, \dots, B_n\}$ in R ,

The *functional dependency* $A \rightarrow B$ on R holds if for *any* t_i, t_j in R :

if $t_i[A_1] = t_j[A_1]$ AND $t_i[A_2] = t_j[A_2]$ AND ... AND $t_i[A_m] = t_j[A_m]$

then $t_i[B_1] = t_j[B_1]$ AND $t_i[B_2] = t_j[B_2]$ AND ... AND $t_i[B_n] = t_j[B_n]$



Contoh

TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

Tentukan FD untuk tabel di atas

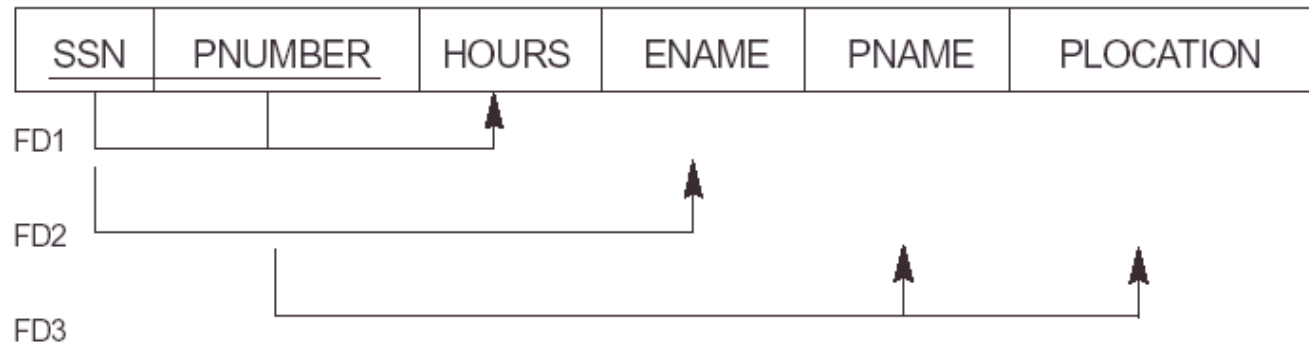


Contoh *Functional Dependency*

EMP_DEPT



EMP_PROJ



Pada relasi EMP_PROJ terdapat 3 FD:

- $\{SSN, PNUMBER\} \rightarrow \{HOURS\}$
- $\{SSN\} \rightarrow \{ENAME\}$
- $\{PNUMBER\} \rightarrow \{PNAME, PLOCATION\}$

Bagaimana menemukan FD lainnya?

Example:

Products

Name	Color	Category	Dep	Price
Gizmo	Green	Gadget	Toys	49
Widget	Black	Gadget	Toys	59
Gizmo	Green	Whatsit	Garden	99

Provided FDs:

1. {Name} \rightarrow {Color}
2. {Category} \rightarrow {Department}
3. {Color, Category} \rightarrow {Price}

Given the provided FDs, we can see that {Name, Category} \rightarrow {Price} must also hold on **any instance**...

Which / how many other FDs do?!?

Menemukan FD lainnya

Sama saja bertanya: Given a set of FDs, $F = \{f_1, \dots, f_n\}$, does an FD x hold?

Inference problem: How do we decide?

Answer: Three simple rules called
Armstrong's Rules.

1. Reflection
2. Augmentation
3. Transitive



Note: Inference Rule

○ *Amstrong's Inference Rule*

1. ***Reflection (IR1):*** If $\{X \supseteq Y\}$ then $X \rightarrow Y$.
2. ***Augmentation (IR2):*** If $\{X \rightarrow Y\}$ then $XZ \rightarrow YZ$.
3. ***Transitive (IR3):*** If $\{X \rightarrow Y, Y \rightarrow Z\}$ then $X \rightarrow Z$.

▶ *Derived Inference Rule*

4. ***Decomposition:*** If $\{X \rightarrow YZ\}$ then $X \rightarrow Y$.
5. ***Additive (Union):*** If $\{X \rightarrow Y, X \rightarrow Z\}$ then $X \rightarrow YZ$.
6. ***Pseudotransitive:*** If $\{X \rightarrow Y, WY \rightarrow Z\}$ then $WX \rightarrow Z$.

Temukan Functional Dependencies

Example:

Inferred FDs:

Inferred FD	Rule used
4. {Name, Category} -> {Name}	?
5. {Name, Category} -> {Color}	?
6. {Name, Category} -> {Category}	?
7. {Name, Category} -> {Color, Category}	?
8. {Name, Category} -> {Price}	?

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Dept.}
3. {Color, Category} → {Price}

Which / how many other FDs hold?



Inference Rule untuk FD

- Diberikan sekumpulan FD F , kita dapat menurunkan FD-FD yang lain dari F
- Contoh:
 $F = \{SSN \rightarrow \{ENAME, BDATE, ADDRESS, DNUMBER\}, \{SSN, PROJNO\} \rightarrow HOURS\}$
- Apa FD yang memiliki makna sama dengan F ?
 - ◆ $SSN \rightarrow ENAME$
 - ◆ $SSN \rightarrow BDATE$
 - ◆ $SSN \rightarrow ADDRESS$
 - ◆ $SSN \rightarrow DNUMBER$
 - ◆ $SSN \rightarrow SSN$
 - ◆ $\{SSN, PROJNO\} \rightarrow HOURS$

Closure

Closure of a set of Attributes

Given a set of attributes A_1, \dots, A_n and a set of FDs F :
Then the closure, $\{A_1, \dots, A_n\}^+$ is the set of attributes B s.t. $\{A_1, \dots, A_n\} \rightarrow B$

Example: $F =$

$$\begin{aligned} \{name\} &\rightarrow \{color\} \\ \{category\} &\rightarrow \{department\} \\ \{color, category\} &\rightarrow \{price\} \end{aligned}$$

Example Closures:

$$\begin{aligned} \{name\}^+ &= \{name, color\} \\ \{name, category\}^+ &= \\ &\{name, category, color, dept, price\} \\ \{color\}^+ &= \{color\} \end{aligned}$$

Closure Algorithm

Start with $X = \{A_1, \dots, A_n\}$ and set of FDs F .

Repeat until X doesn't change; do:

if $\{B_1, \dots, B_n\} \rightarrow C$ is entailed by F

and $\{B_1, \dots, B_n\} \subseteq X$

then add C to X .

Return X as X^+

Closure Algorithm

Start with $X = \{A_1, \dots, A_n\}$, FDs F .
Repeat until X doesn't change; do:
 if $\{B_1, \dots, B_n\} \rightarrow C$ is in F and $\{B_1, \dots, B_n\} \subseteq X$:
 then add C to X .
Return X as X^+

$\{\text{name, category}\}^+ =$
 $\{\text{name, category}\}$

$F =$

$\{\text{name}\} \rightarrow \{\text{color}\}$

$\{\text{category}\} \rightarrow \{\text{dept}\}$

$\{\text{color, category}\} \rightarrow \{\text{price}\}$

Closure Algorithm

Start with $X = \{A_1, \dots, A_n\}$, FDs F .
Repeat until X doesn't change; do:
 if $\{B_1, \dots, B_n\} \rightarrow C$ is in F and $\{B_1, \dots, B_n\} \subseteq X$:
 then add C to X .
Return X as X^+

$\{\text{name}, \text{category}\}^+ =$
 $\{\text{name}, \text{category}\}$

$\{\text{name}, \text{category}\}^+ =$
 $\{\text{name}, \text{category}, \text{color}\}$

$F =$

$\{\text{name}\} \rightarrow \{\text{color}\}$

$\{\text{category}\} \rightarrow \{\text{dept}\}$

$\{\text{color}, \text{category}\} \rightarrow \{\text{price}\}$

Closure Algorithm

Start with $X = \{A_1, \dots, A_n\}$, FDs F .
Repeat until X doesn't change; do:
 if $\{B_1, \dots, B_n\} \rightarrow C$ is in F and $\{B_1, \dots, B_n\} \subseteq X$:
 then add C to X .
Return X as X^+

$F =$

$\{\text{name}\} \rightarrow \{\text{color}\}$

$\{\text{category}\} \rightarrow \{\text{dept}\}$

$\{\text{color, category}\} \rightarrow \{\text{price}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category, color}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category, color, dept}\}$

Closure Algorithm

Start with $X = \{A_1, \dots, A_n\}$, FDs F .
Repeat until X doesn't change; do:
 if $\{B_1, \dots, B_n\} \rightarrow C$ is in F and $\{B_1, \dots, B_n\} \subseteq X$:
 then add C to X .
Return X as X^+

$F =$

$\{\text{name}\} \rightarrow \{\text{color}\}$

$\{\text{category}\} \rightarrow \{\text{dept}\}$

$\{\text{color, category}\} \rightarrow \{\text{price}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category, color}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category, color, dept}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category, color, dept, price}\}$

Closure Algorithm

Start with $X = \{A_1, \dots, A_n\}$, FDs F .
Repeat until X doesn't change; do:
 if $\{B_1, \dots, B_n\} \rightarrow C$ is in F and $\{B_1, \dots, B_n\} \subseteq X$:
 then add C to X .
Return X as X^+

$F =$

$\{\text{name}\} \rightarrow \{\text{color}\}$

$\{\text{category}\} \rightarrow \{\text{dept}\}$

$\{\text{color, category}\} \rightarrow \{\text{price}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category, color, dept, price}\}$

Additional FDs:

$\{\text{name, category}\} \rightarrow \{\text{color}\}$

$\{\text{name, category}\} \rightarrow \{\text{dept}\}$

$\{\text{name, category}\} \rightarrow \{\text{price}\}$

Example

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

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

$\{A,D\} \rightarrow \{E\}$

$\{B\} \rightarrow \{D\}$

$\{A,F\} \rightarrow \{B\}$

Compute $\{A,B\}^+ = \{A, B, \quad \}$

Compute $\{A, F\}^+ = \{A, F, \quad \}$

Kenapa kita membutuhkan *closure*?

- With closure we can find all FD's easily
- To check if $X \rightarrow A$

1. Compute X^+

2. Check if $A \in X^+$

Note here that X is a set of attributes, but A is a *single* attribute. Why does considering FDs of this form suffice?

Recall the Split/combine rule:

$X \rightarrow A_1, \dots, X \rightarrow A_n$

implies

$X \rightarrow \{A_1, \dots, A_n\}$

Kenapa kita membutuhkan *closure*?

- For each set of attributes X
 1. Compute X^+
 2. If $X^+ =$ set of all attributes then X is a **superkey**
 3. If X is minimal, then it is a **key**

Temukan *key*-nya

Product(name, price, category, color)

{name, category} → price
{category} → color

What is a key?

Normalisasi



Normalisasi

- Normalisasi (Normalization):
 - Proses dekomposisi relasi yang masih “buruk” dengan memecah atribut-atributnya untuk membentuk beberapa relasi
- Bentuk Normal (Normal Form)
 - Kondisi (dengan menggunakan FD dan key) yang menentukan apakah suatu skema relasi memenuhi kriteria tertentu



Key Attributes

- **Superkey dari relasi R**: himpunan attribute dari R yang dapat membedakan satu tuple dengan tuple lainnya pada R.
- **Key K**: Superkey minimal, sedemikian hingga penghapusan salah satu attribute dari K akan menyebabkan K tidak lagi menjadi *key*
- Jika relasi punya beberapa key, salah satu dari *key* dipilih menjadi **primary key**, sedangkan yang lain menjadi *candidate key*.
- **Prime attribute**: Sebuah attribute yang menjadi anggota dari *key*
- **Non prime attribute**: Sebuah attribute yang bukan merupakan anggota *key* manapun.



Bentuk Normal Pertama (1NF)

- Bentuk normal pertama (1NF) merupakan bagian dari definisi relasi.
- Bentuk normal pertama tidak mengizinkan:
 - *Composite attributes,*
 - *Multivalued attributes,*
 - *Nested relations.*



Normalisasi dari Multivalued ke 1NF

(a)

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn	Dlocations

(b)

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn	<u>Dlocation</u>
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Figure 10.8

Normalization into 1NF.

(a) A relation schema that is not in 1NF. (b) Example state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.



Normalisasi dari Nested Relation ke 1NF

(a)

EMP_PROJ		Projs	
Ssn	Ename	Pnumber	Hours

(b)

Ssn	Ename	Pnumber	Hours
123456789	Smith, John B.	1	32.5
		2	7.5
666884444	Narayan, Ramesh K.	3	40.0
453453453	English, Joyce A.	1	20.0
		2	20.0
333445555	Wong, Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya, AliciaJ.	30	30.0
		10	10.0
987987987	Jabbar, Ahmad V.	10	35.0
		30	5.0
987654321	Wallace, Jennifer S.	30	20.0
		20	15.0
888665555	Borg, James E.	20	NULL

(c)

EMP_PROJ1

Ssn	Ename
-----	-------

EMP_PROJ2

Ssn	Pnumber	Hours
-----	---------	-------

Figure 10.9

Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a *nested relation* attribute PROJS. (b) Example extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposition of EMP_PROJ into relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.



Bentuk Normal Kedua (2NF)

- Menggunakan konsep FD dan *key*
- Definisi:
 - *Full functional dependency*: suatu FD $X \rightarrow Y$ sedemikian hingga jika salah satu attribute dari X dibuang, maka FD tersebut tidak ada lagi.
- Contoh:
 - $\{SSN, PNUMBER\} \rightarrow HOURS$ merupakan full FD, karena tidak ada $SSN \rightarrow HOURS$ dan $PNUMBER \rightarrow HOURS$
 - $\{SSN, PNUMBER\} \rightarrow ENAME$ merupakan *partial dependency*, bukan full FD karena terdapat dependency $SSN \rightarrow ENAME$

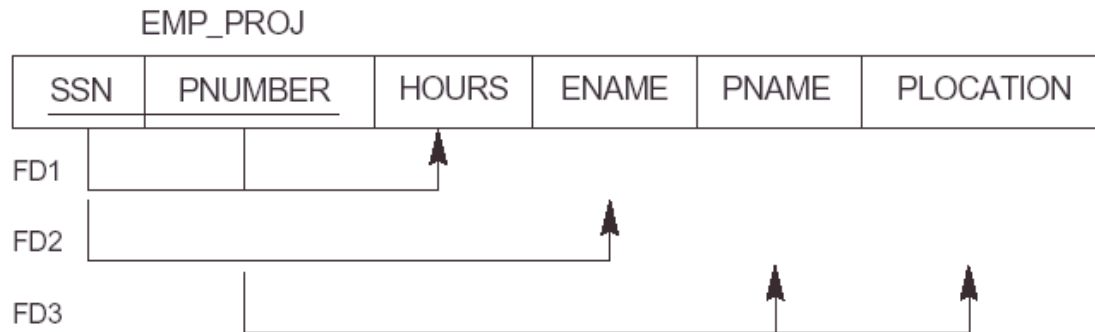


Bentuk Normal Kedua (2NF)

- Suatu relasi R berada dalam bentuk normal kedua (2NF) jika R berada dalam 1NF **dan**
- Skema relasi R berada pada 2NF (general definition) jika tidak ada *nonprime attribute A* pada R *partially dependent* pada **sebuah candidate key**

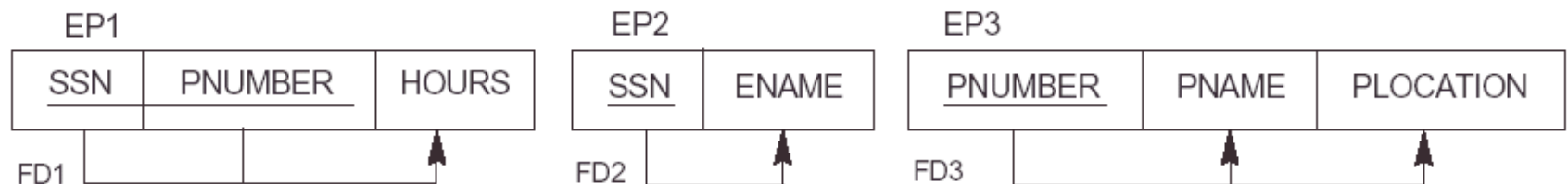
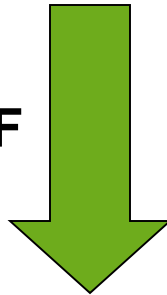


Contoh Normalisasi 2NF



- **ENAME fully dependent on SSN**
- **PNAME, PLOCATION fully dependent on PNUMBER**

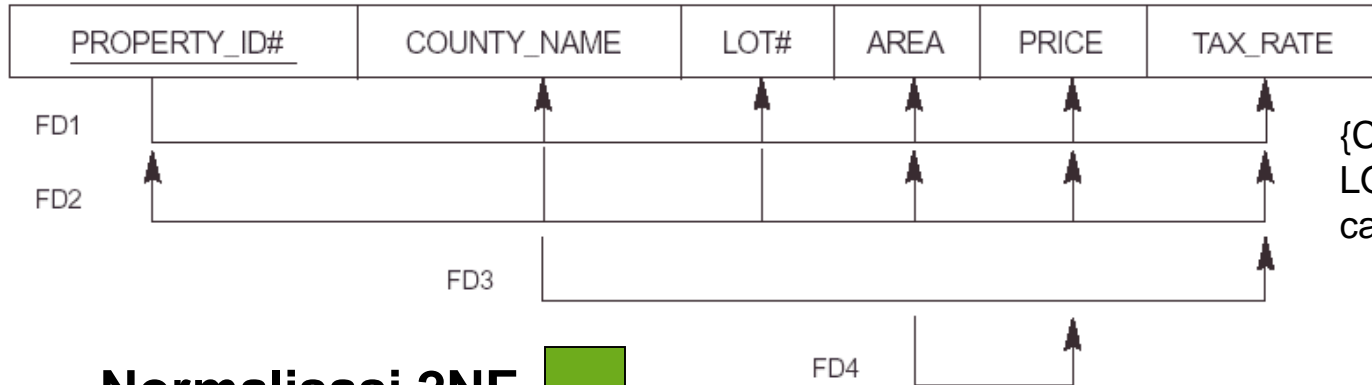
Normalisasi 2NF





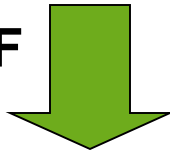
Contoh Normalisasi 2NF

LOTS

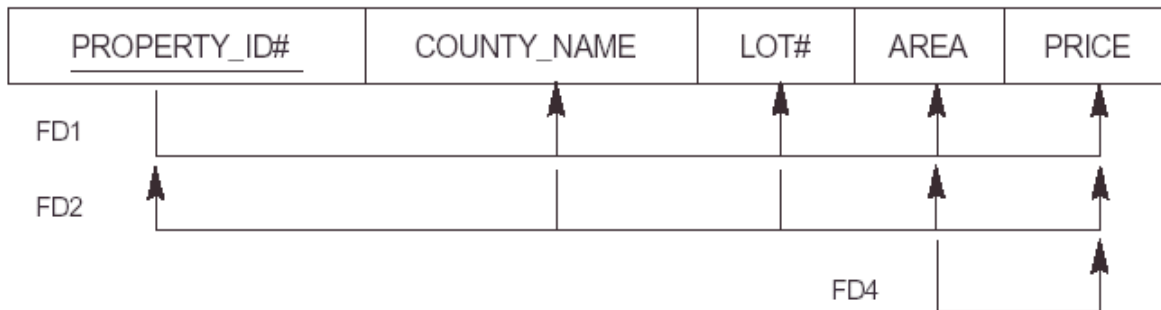


{COUNTY_NAME, LOT#} is candidate key

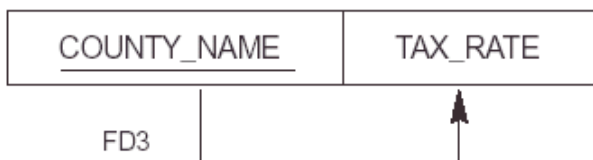
Normalisasi 2NF



LOTS1



LOTS2





Bentuk Normal Ketiga (3NF)

- Suatu relasi R berada dalam bentuk normal ketiga (3NF) jika R berada dalam 2NF **dan**
- Skema relasi R berada dalam 3NF dengan syarat jika terdapat FD $X \rightarrow Y$ maka:
 - (a) X merupakan **sebuah** superkey dari R **atau**
 - (b) Y merupakan prime attribute dari R

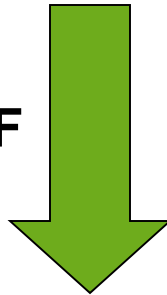


Contoh Normalisasi 3NF

EMP_DEPT



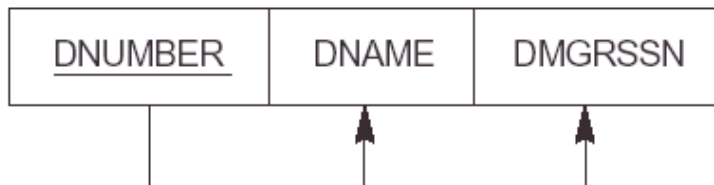
Normalisasi 3NF



ED1



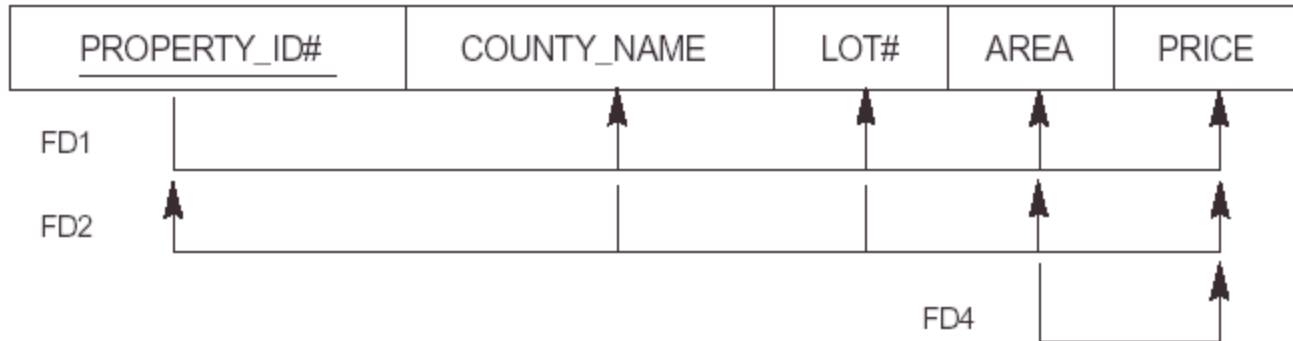
ED2



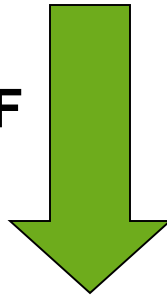


Contoh Normalisasi 3NF

LOTS1



Normalisasi 3NF





Contoh Normalisasi 3NF

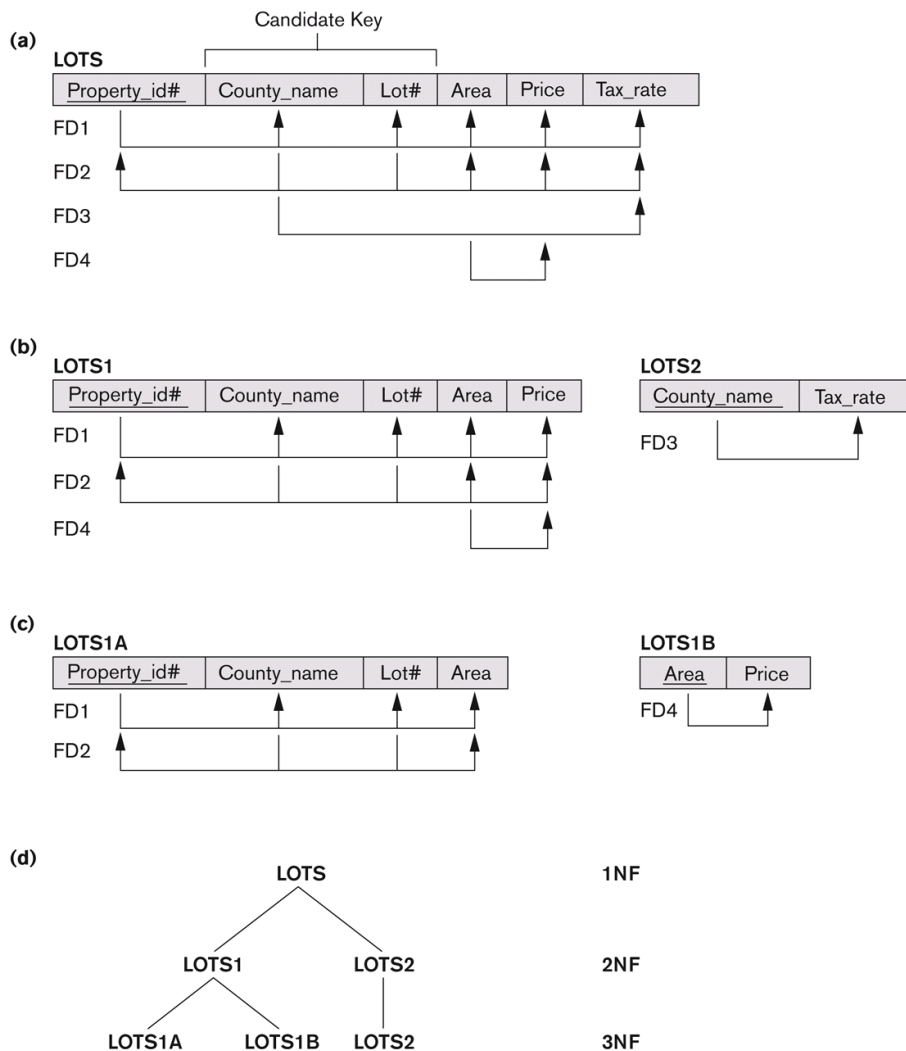


Figure 10.11

Normalization into 2NF and 3NF. (a) The LOTS relation with its functional dependencies FD1 through FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Summary of the progressive normalization of LOTS.

Boyce Codd Normal Form



Definisi dalam General Normal Form

- Skema relasi R berada dalam 3NF dengan syarat jika terdapat FD $X \rightarrow Y$ maka:
 - (a) X merupakan **sebuah** superkey dari R atau
 - (b) Y merupakan **prime attribute** dari R
- BCNF **tidak membolehkan kondisi (b)** di atas

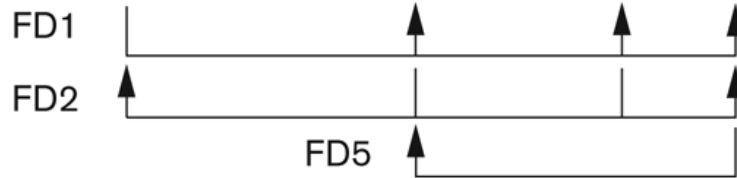


Contoh BCNF

(a)

LOTS1A

<u>Property_id#</u>	County_name	Lot#	Area
---------------------	-------------	------	------



BCNF Normalization

LOTS1AX

<u>Property_id#</u>	Area	Lot#
---------------------	------	------

LOTS1AY

<u>Area</u>	County_name
-------------	-------------

(b)

R

<u>A</u>	<u>B</u>	C
----------	----------	---

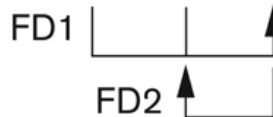


Figure 10.12

Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF.



Contoh: Relasi pada 3NF Namun tidak BCNF

TEACH

Student	Course	Instructor
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe
Narayan	Operating Systems	Ammar

Figure 10.13

A relation TEACH that
is in 3NF but not
BCNF.



Normalisasi BCNF

- Ada 2 FD pada relasi TEACH:
 - fd1: { student, course} \rightarrow instructor
 - fd2: instructor \rightarrow course
- {student, course} merupakan candidate key untuk teach
 - Relasi ini berada pada 3NF *namun tidak pada* BCNF
- Relasi yang belum BCNF dapat didekomposisi untuk mencapai BCNF, namun kadang-kadang dapat menghilangkan FD yang semula telah ada



Normalisasi BCNF

- Ada 3 dekomposisi yang mungkin untuk relasi TEACH:
 - {student, instructor} and {student, course}
 - {course, instructor} and {course, student}
 - {instructor, course} and {instructor, student}
- Semua dekomposisi tersebut kehilangan FD1
 - Semua FD yang ada pada relasi sedapat mungkin dijaga, namun dapat dimaklumi jika hilang pada normalisasi BCF
 - Namun sifat lossless dan non-additivity property setelah dekomposisi harus tetap dijaga



Referensi

- Elmasri & Navathe, Fundamental of Database Systems, 7th Edition, Chapter 15, 2015