

Lecture 22: Reasoning about Functional Dependencies

More on FD's and Normal Forms

- How not to be in 1NF
- Revision of constraints and normal forms
- stronger versions of 2NF, 3NF from [E&N]
- Boyce-Codd Normal Form (BCNF)

See [E&N Chapter 10], especially the second half.

Announcements

- A good alternate source for the normal forms material is Carol Edmonson's 2006 lecture slides. There is a link on the lectures page.

First Normal Form

[E&N §10.3.4]

Recall that a relation is a subset of a set-product. Therefore, it is a set of tuples.

First Normal Form

[E&N §10.3.4]

Recall that a relation is a subset of a set-product. Therefore, it is a set of tuples.

A relation is in *First Normal Form* (1NF) if the values in those tuples are *atomic*, that is, not tuples or sets.

First Normal Form

[E&N §10.3.4]

Recall that a relation is a subset of a set-product. Therefore, it is a set of tuples.

A relation is in *First Normal Form* (1NF) if the values in those tuples are *atomic*, that is, not tuples or sets.

These two relations are not in 1NF

$$\{(London, (1, 2, 3), herring)\}$$
$$\{(Paris, \{4, 5, 6\}, kipper)\}$$

Products and Products

One way that non-1NF relations might arise:

- we have had different, even deliberately ambiguous definitions of product

Products and Products

One way that non-1NF relations might arise:

- we have had different, even deliberately ambiguous definitions of product
- we want to use both qualified names *employee.ssn* and unqualified names *deptID* to access the same product

Products and Products

One way that non-1NF relations might arise:

- we have had different, even deliberately ambiguous definitions of product
- we want to use both qualified names *employee.ssn* and unqualified names *deptID* to access the same product
- so does *employee* \times *department* contain pairs of tuples, or big tuples with attributes from both input relations?

Maths for Real and Tuple Valued Attributes

Using the strict mathematical definition of product with database relations yields a relation with attributes whose values are tuples.

$$\begin{aligned} \mathbf{R} &= (\mathbf{A} = \{\{\mathbf{a} = 1, \mathbf{x} = 2\}\}) \times (\mathbf{B} = \{\{\mathbf{b} = 3, \mathbf{y} = 4\}\}) \\ &= \{\{\mathbf{A} = \{\mathbf{a} = 1, \mathbf{x} = 2\}, \mathbf{B} = \{\mathbf{b} = 3, \mathbf{y} = 4\}\}\} \end{aligned}$$

Maths for Real and Tuple Valued Attributes

Using the strict mathematical definition of product with database relations yields a relation with attributes whose values are tuples.

$$\begin{aligned} \mathbf{R} &= (\mathbf{A} = \{\{\mathbf{a} = 1, \mathbf{x} = 2\}\}) \times (\mathbf{B} = \{\{\mathbf{b} = 3, \mathbf{y} = 4\}\}) \\ &= \{\{\mathbf{A} = \{\mathbf{a} = 1, \mathbf{x} = 2\}, \mathbf{B} = \{\mathbf{b} = 3, \mathbf{y} = 4\}\}\} \end{aligned}$$

In tabular form

R			
A		B	
a	x	b	y
1	2	3	4

Database Theory, Flat Tuples and Aliases

To keep relations in 1NF, product in database theory works like this

$$\begin{aligned} \mathbf{R} &= (\mathbf{A} = \{\{\mathbf{a} = 1, \mathbf{x} = 2\}\}) \times (\mathbf{B} = \{\{\mathbf{b} = 3, \mathbf{y} = 4\}\}) \\ &= \{\{\mathbf{a} = \mathbf{A.a} = 1, \mathbf{x} = \mathbf{A.x} = 2, \mathbf{b} = \mathbf{B.b} = 3, \mathbf{y} = \mathbf{B.y} = 4\}\} \end{aligned}$$

Database Theory, Flat Tuples and Aliases

To keep relations in 1NF, product in database theory works like this

$$\begin{aligned} \mathbf{R} &= (\mathbf{A} = \{\{\mathbf{a} = 1, \mathbf{x} = 2\}\}) \times (\mathbf{B} = \{\{\mathbf{b} = 3, \mathbf{y} = 4\}\}) \\ &= \{\{\mathbf{a} = \mathbf{A.a} = 1, \mathbf{x} = \mathbf{A.x} = 2, \mathbf{b} = \mathbf{B.b} = 3, \mathbf{y} = \mathbf{B.y} = 4\}\} \end{aligned}$$

Which we can write as the following table

a=A.a	x=A.x	b=B.b	y=B.y
1	2	3	4

First Normalisation - Structured

Flattening a structured relation is straight-forward
(rename attributes to avoid collisions)

city	ns	fish
London	(1,2,3)	herring

First Normalisation - Structured

Flattening a structured relation is straight-forward
(rename attributes to avoid collisions)

city	ns	fish
London	(1,2,3)	herring

becomes

city	n1	n2	n3	fish
London	1	2	3	herring

First Normalisation - Set-Valued

If you have a set-valued attribute, create a new table with

- foreign key attribute(s) referencing the original primary key

First Normalisation - Set-Valued

If you have a set-valued attribute, create a new table with

- foreign key attribute(s) referencing the original primary key
- single valued *attrib* : *type* replacing original *attrib* : *type set*

First Normalisation - Set-Valued

If you have a set-valued attribute, create a new table with

- foreign key attribute(s) referencing the original primary key
- single valued attrib : *type* replacing original *attrib* : *type set*
- both attribs must be key in new table

First Normalisation - Set-Valued

If you have a set-valued attribute, create a new table with

- foreign key attribute(s) referencing the original primary key
- single valued attrib : *type* replacing original *attrib* : *type set*
- both attribs must be key in new table

First Normalisation - Set-Valued

If you have a set-valued attribute, create a new table with

- foreign key attribute(s) referencing the original primary key
- single valued attrib : *type* replacing original *attrib* : *type set*
- both attribs must be key in new table

You can use the same technique for translating UML or ER multi-valued attributes to single-valued relational attributes.

First Normalisation - Set-Valued - Example

city	numbers	fish
Paris	{4,5,6}	kipper
New York	{7}	anchovy

First Normalisation - Set-Valued - Example

<u>city</u>	numbers	fish
Paris	{4,5,6}	kipper
New York	{7}	anchovy



baseTable

<u>city</u>	fish
Paris	kipper
New York	anchovy

attribTable

<u>city</u>	<u>number</u>
Paris	4
Paris	5
Paris	6
New York	7

attribTable(city) references *baseTable(city)*

Constraints Revision: FD and Superkey

Notation: We write \mathbf{R} for the set of attributes of the relation schema of \mathbf{R} .

Constraints Revision: FD and Superkey

Notation: We write \mathbf{R} for the set of attributes of the relation schema of \mathbf{R} .

Functional dependency $X \longrightarrow Y$

- $X, Y \subseteq \mathbf{R}$ are sets of attributes

Constraints Revision: FD and Superkey

Notation: We write \mathbf{R} for the set of attributes of the relation schema of \mathbf{R} .

Functional dependency $X \longrightarrow Y$

- $X, Y \subseteq \mathbf{R}$ are sets of attributes
- $X \longrightarrow Y$ holds in a relation R of \mathbf{R} iff
$$\pi_X(t_1) = \pi_X(t_2) \Rightarrow \pi_Y(t_1) = \pi_Y(t_2)$$

Constraints Revision: FD and Superkey

Notation: We write \mathbf{R} for the set of attributes of the relation schema of \mathbf{R} .

Functional dependency $X \longrightarrow Y$

- $X, Y \subseteq \mathbf{R}$ are sets of attributes
- $X \longrightarrow Y$ holds in a relation R of \mathbf{R} iff
$$\pi_X(t_1) = \pi_X(t_2) \Rightarrow \pi_Y(t_1) = \pi_Y(t_2)$$
- ie, same X values, same Y values

Constraints Revision: FD and Superkey

Notation: We write \mathbf{R} for the set of attributes of the relation schema of \mathbf{R} .

Functional dependency $X \longrightarrow Y$

- $X, Y \subseteq \mathbf{R}$ are sets of attributes
- $X \longrightarrow Y$ holds in a relation R of \mathbf{R} iff $\pi_X(t_1) = \pi_X(t_2) \Rightarrow \pi_Y(t_1) = \pi_Y(t_2)$
- ie, same X values, same Y values
- “ X is a superkey” means $X \longrightarrow \mathbf{R}$.

Constraints Revision: FD and Superkey

Notation: We write \mathbf{R} for the set of attributes of the relation schema of \mathbf{R} .

Functional dependency $X \longrightarrow Y$

- $X, Y \subseteq \mathbf{R}$ are sets of attributes
- $X \longrightarrow Y$ holds in a relation R of \mathbf{R} iff $\pi_X(t_1) = \pi_X(t_2) \Rightarrow \pi_Y(t_1) = \pi_Y(t_2)$
- ie, same X values, same Y values
- “ X is a superkey” means $X \longrightarrow \mathbf{R}$.
- ie, $\pi_X(t_1) = \pi_X(t_2) \Rightarrow t_1 = t_2$

Constraints Revision: FD and Superkey

Notation: We write \mathbf{R} for the set of attributes of the relation schema of \mathbf{R} .

Functional dependency $X \longrightarrow Y$

- $X, Y \subseteq \mathbf{R}$ are sets of attributes
- $X \longrightarrow Y$ holds in a relation R of \mathbf{R} iff $\pi_X(t_1) = \pi_X(t_2) \Rightarrow \pi_Y(t_1) = \pi_Y(t_2)$
- ie, same X values, same Y values
- “ X is a superkey” means $X \longrightarrow \mathbf{R}$.
- ie, $\pi_X(t_1) = \pi_X(t_2) \Rightarrow t_1 = t_2$

Constraints Revision: FD and Superkey

Notation: We write \mathbf{R} for the set of attributes of the relation schema of \mathbf{R} .

Functional dependency $X \longrightarrow Y$

- $X, Y \subseteq \mathbf{R}$ are sets of attributes
- $X \longrightarrow Y$ holds in a relation R of \mathbf{R} iff $\pi_X(t_1) = \pi_X(t_2) \Rightarrow \pi_Y(t_1) = \pi_Y(t_2)$
- ie, same X values, same Y values
- “ X is a superkey” means $X \longrightarrow \mathbf{R}$.
- ie, $\pi_X(t_1) = \pi_X(t_2) \Rightarrow t_1 = t_2$

What is $\pi_{\mathbf{R}}(t)$?

Constraints Revision: FD and Superkey

Notation: We write \mathbf{R} for the set of attributes of the relation schema of \mathbf{R} .

Functional dependency $X \longrightarrow Y$

- $X, Y \subseteq \mathbf{R}$ are sets of attributes
- $X \longrightarrow Y$ holds in a relation R of \mathbf{R} iff $\pi_X(t_1) = \pi_X(t_2) \Rightarrow \pi_Y(t_1) = \pi_Y(t_2)$
- ie, same X values, same Y values
- “ X is a superkey” means $X \longrightarrow \mathbf{R}$.
- ie, $\pi_X(t_1) = \pi_X(t_2) \Rightarrow t_1 = t_2$

What is $\pi_{\mathbf{R}}(t)$? $\pi_{\mathbf{R}}(t) = t$

Constraints Revision

- $X \longrightarrow Y$ is a *full* functional dependency

Constraints Revision

- $X \longrightarrow Y$ is a *full* functional dependency
 - Y depends on *all* of X

Constraints Revision

- $X \longrightarrow Y$ is a *full* functional dependency
 - Y depends on *all* of X
 - X is minimal (small as it can be while still determining Y)

Constraints Revision

- $X \longrightarrow Y$ is a *full* functional dependency
 - Y depends on *all* of X
 - X is minimal (small as it can be while still determining Y)
 - there is no $X' \subset X$ ($X' \neq X$) where $X' \longrightarrow Y$

Constraints Revision

- $X \longrightarrow Y$ is a *full* functional dependency
 - Y depends on *all* of X
 - X is minimal (small as it can be while still determining Y)
 - there is no $X' \subset X$ ($X' \neq X$) where $X' \longrightarrow Y$
- X is a key (or candidate key)

Constraints Revision

- $X \longrightarrow Y$ is a *full* functional dependency
 - Y depends on *all* of X
 - X is minimal (small as it can be while still determining Y)
 - there is no $X' \subset X$ ($X' \neq X$) where $X' \longrightarrow Y$
- X is a key (or candidate key)
 - if it is a minimal superkey

Constraints Revision

- $X \longrightarrow Y$ is a *full* functional dependency
 - Y depends on *all* of X
 - X is minimal (small as it can be while still determining Y)
 - there is no $X' \subset X$ ($X' \neq X$) where $X' \longrightarrow Y$
- X is a key (or candidate key)
 - if it is a minimal superkey
 - X is a superkey *and* no $X' \subset X$ ($X' \neq X$) is a superkey

Constraints Revision

- $X \longrightarrow Y$ is a *full* functional dependency
 - Y depends on *all* of X
 - X is minimal (small as it can be while still determining Y)
 - there is no $X' \subset X$ ($X' \neq X$) where $X' \longrightarrow Y$
- X is a key (or candidate key)
 - if it is a minimal superkey
 - X is a superkey *and* no $X' \subset X$ ($X' \neq X$) is a superkey
 - $X \longrightarrow \mathbf{R}$ is a full functional dependency

Constraints Revision

- $X \longrightarrow Y$ is a *full* functional dependency
 - Y depends on *all* of X
 - X is minimal (small as it can be while still determining Y)
 - there is no $X' \subset X$ ($X' \neq X$) where $X' \longrightarrow Y$
- X is a key (or candidate key)
 - if it is a minimal superkey
 - X is a superkey *and* no $X' \subset X$ ($X' \neq X$) is a superkey
 - $X \longrightarrow \mathbf{R}$ is a full functional dependency
- A is a *prime* attribute if it is in a key

Constraints Revision

- $X \longrightarrow Y$ is a *full* functional dependency
 - Y depends on *all* of X
 - X is minimal (small as it can be while still determining Y)
 - there is no $X' \subset X$ ($X' \neq X$) where $X' \longrightarrow Y$
- X is a key (or candidate key)
 - if it is a minimal superkey
 - X is a superkey *and* no $X' \subset X$ ($X' \neq X$) is a superkey
 - $X \longrightarrow \mathbf{R}$ is a full functional dependency
- A is a *prime* attribute if it is in a key
 - that's *any* key, not just the primary key

Constraints Revision

- $X \longrightarrow Y$ is a *full* functional dependency
 - Y depends on *all* of X
 - X is minimal (small as it can be while still determining Y)
 - there is no $X' \subset X$ ($X' \neq X$) where $X' \longrightarrow Y$
- X is a key (or candidate key)
 - if it is a minimal superkey
 - X is a superkey *and* no $X' \subset X$ ($X' \neq X$) is a superkey
 - $X \longrightarrow \mathbf{R}$ is a full functional dependency
- A is a *prime* attribute if it is in a key
 - that's *any* key, not just the primary key
 - a *non-prime* attribute is not in any candidate key

Non-Transitive Dependency Non-Revision

Non-transitive dependency (used in 3NF) was not carefully defined last time, but needs to be.

Non-Transitive Dependency Non-Revision

Non-transitive dependency (used in 3NF) was not carefully defined last time, but needs to be.

$X \longrightarrow Y$ in \mathbf{R} is a “*transitive dependency* if there is a set of attributes Z that is neither a candidate key nor a subset of any key of \mathbf{R} , and both $X \longrightarrow Z$ and $Z \longrightarrow Y$ hold.” [E&N §10.3.6]

Non-Transitive Dependency Non-Revision

Non-transitive dependency (used in 3NF) was not carefully defined last time, but needs to be.

$X \longrightarrow Y$ in \mathbf{R} is a “*transitive dependency* if there is a set of attributes Z that is neither a candidate key nor a subset of any key of \mathbf{R} , and both $X \longrightarrow Z$ and $Z \longrightarrow Y$ hold.” [E&N §10.3.6]

ie, the “middle-man” can’t be in a key

Non-Transitive Dependency Non-Revision

Non-transitive dependency (used in 3NF) was not carefully defined last time, but needs to be.

$X \longrightarrow Y$ in \mathbf{R} is a “*transitive dependency* if there is a set of attributes Z that is neither a candidate key nor a subset of any key of \mathbf{R} , and both $X \longrightarrow Z$ and $Z \longrightarrow Y$ hold.” [E&N §10.3.6]

ie, the “middle-man” can’t be in a key

And (they forgot to mention) $Y \not\subseteq Z$. ie $Z \longrightarrow Y$ is non-trivial.

Non-Transitive Dependency Non-Revision

Non-transitive dependency (used in 3NF) was not carefully defined last time, but needs to be.

$X \longrightarrow Y$ in \mathbf{R} is a “*transitive dependency* if there is a set of attributes Z that is neither a candidate key nor a subset of any key of \mathbf{R} , and both $X \longrightarrow Z$ and $Z \longrightarrow Y$ hold.” [E&N §10.3.6]

ie, the “middle-man” can’t be in a key

And (they forgot to mention) $Y \not\subseteq Z$. ie $Z \longrightarrow Y$ is non-trivial.

Y is *non-transitively dependent* on X if $X \longrightarrow Y$, holds but is not a transitive dependency.

Normal Form Revision

Remember, each normal form is a property a relation schema \mathbf{R} can have, *relative to a given set of functional dependencies F .*

Normal Form Revision

Remember, each normal form is a property a relation schema \mathbf{R} can have, *relative to a given set of functional dependencies F .*

1NF attribute values are “atomic”, not tuples, not sets

Normal Form Revision

Remember, each normal form is a property a relation schema \mathbf{R} can have, *relative to a given set of functional dependencies F* .

1NF attribute values are “atomic”, not tuples, not sets

2NF 1NF & every non-prime attribute is fully functionally dependent on the primary key

Normal Form Revision

Remember, each normal form is a property a relation schema R can have, *relative to a given set of functional dependencies F* .

- 1NF attribute values are “atomic”, not tuples, not sets
- 2NF 1NF & every non-prime attribute is fully functionally dependent on the primary key
- 3NF 2NF & every non-prime attribute is non-transitively dependent on the primary key

Normal Form Revision

Remember, each normal form is a property a relation schema R can have, *relative to a given set of functional dependencies F* .

- 1NF attribute values are “atomic”, not tuples, not sets
- 2NF 1NF & every non-prime attribute is fully functionally dependent on the primary key
- 3NF 2NF & every non-prime attribute is non-transitively dependent on the primary key

Normal Form Revision

Remember, each normal form is a property a relation schema R can have, *relative to a given set of functional dependencies F .*

- 1NF attribute values are “atomic”, not tuples, not sets
- 2NF 1NF & every non-prime attribute is fully functionally dependent on the primary key
- 3NF 2NF & every non-prime attribute is non-transitively dependent on the primary key

Note that any attributes in candidate keys other than the primary key are not involved at all in 2NF and 3NF.

Stronger forms of 2 and 3NF

In [E&N §10.4], stronger, “general” forms of 2NF and 3NF are introduced. I will call them EN2NF, EN3NF respectively.

Stronger forms of 2 and 3NF

In [E&N §10.4], stronger, “general” forms of 2NF and 3NF are introduced. I will call them EN2NF, EN3NF respectively.

“Stronger” means, any relation in EN2NF is in 2NF, but not necessarily vice-versa.

EN2NF 1NF & every non-prime attribute is fully functionally dependent on every key

Stronger forms of 2 and 3NF

In [E&N §10.4], stronger, “general” forms of 2NF and 3NF are introduced. I will call them EN2NF, EN3NF respectively.

“Stronger” means, any relation in EN2NF is in 2NF, but not necessarily vice-versa.

EN2NF 1NF & every non-prime attribute is fully functionally dependent on *every* key

EN3NF EN2NF & every non-prime attribute is non-transitively dependent on *every* key

Stronger forms of 2 and 3NF

In [E&N §10.4], stronger, “general” forms of 2NF and 3NF are introduced. I will call them EN2NF, EN3NF respectively.

“Stronger” means, any relation in EN2NF is in 2NF, but not necessarily vice-versa.

EN2NF 1NF & every non-prime attribute is fully functionally dependent on *every* key

EN3NF EN2NF & every non-prime attribute is non-transitively dependent on *every* key

Stronger forms of 2 and 3NF

In [E&N §10.4], stronger, “general” forms of 2NF and 3NF are introduced. I will call them EN2NF, EN3NF respectively.

“Stronger” means, any relation in EN2NF is in 2NF, but not necessarily vice-versa.

EN2NF 1NF & every non-prime attribute is fully functionally dependent on *every* key

EN3NF EN2NF & every non-prime attribute is non-transitively dependent on *every* key

That is, R is in EN3NF wrt F iff every non-prime attribute is fully and non-transitively dependent on every key

Another form of EN3NF

Its easier to see the connection with BCNF if we use a different formulation of EN3NF.

- 1 every non-prime attribute is fully and non-transitively dependent on every key

Another form of EN3NF

Its easier to see the connection with BCNF if we use a different formulation of EN3NF.

- 1 every non-prime attribute is fully and non-transitively dependent on every key
- 2 whenever a non-trivial $X \longrightarrow A$ holds, either X is a superkey or A is prime

Another form of EN3NF

Its easier to see the connection with BCNF if we use a different formulation of EN3NF.

- 1 every non-prime attribute is fully and non-transitively dependent on every key
- 2 whenever a non-trivial $X \longrightarrow A$ holds, either X is a superkey or A is prime

Another form of EN3NF

Its easier to see the connection with BCNF if we use a different formulation of EN3NF.

- 1 every non-prime attribute is fully and non-transitively dependent on every key
- 2 whenever a non-trivial $X \longrightarrow A$ holds, either X is a superkey or A is prime

$X \longrightarrow A$ is *trivial* if $A \in X$

Another form of EN3NF

Its easier to see the connection with BCNF if we use a different formulation of EN3NF.

- 1 every non-prime attribute is fully and non-transitively dependent on every key
- 2 whenever a non-trivial $X \longrightarrow A$ holds, either X is a superkey or A is prime

$X \longrightarrow A$ is *trivial* if $A \in X$

(2) means that the only things that non-trivially determine non-primes are superkeys

Another form of EN3NF

Its easier to see the connection with BCNF if we use a different formulation of EN3NF.

- 1 every non-prime attribute is fully and non-transitively dependent on every key
- 2 whenever a non-trivial $X \longrightarrow A$ holds, either X is a superkey or A is prime

$X \longrightarrow A$ is *trivial* if $A \in X$

(2) means that the only things that non-trivially determine non-primes are superkeys

The two definitions are equivalent ...

EN3NF.1 \Rightarrow *EN3NF.2*

Assume that every non-prime attribute is fully and non-transitively dependent on every key.

EN3NF.1 \Rightarrow *EN3NF.2*

Assume that every non-prime attribute is fully and non-transitively dependent on every key.

And assume that $X \longrightarrow A$ and $A \notin X$, and A is non-prime. We now need to prove X is a superkey.

EN3NF.1 \Rightarrow *EN3NF.2*

Assume that every non-prime attribute is fully and non-transitively dependent on every key.

And assume that $X \longrightarrow A$ and $A \notin X$, and A is non-prime. We now need to prove X is a superkey.

Every key K determines everything, including X ,
Therefore $K \longrightarrow X \longrightarrow A$.

EN3NF.1 \Rightarrow *EN3NF.2*

Assume that every non-prime attribute is fully and non-transitively dependent on every key.

And assume that $X \longrightarrow A$ and $A \notin X$, and A is non-prime. We now need to prove X is a superkey.

Every key K determines everything, including X ,
Therefore $K \longrightarrow X \longrightarrow A$.

X must be a key, or a proper subset of a key, otherwise A is transitively dependent on K , contradiction.

EN3NF.1 \Rightarrow EN3NF.2

Assume that every non-prime attribute is fully and non-transitively dependent on every key.

And assume that $X \longrightarrow A$ and $A \notin X$, and A is non-prime. We now need to prove X is a superkey.

Every key K determines everything, including X ,
Therefore $K \longrightarrow X \longrightarrow A$.

X must be a key, or a proper subset of a key, otherwise A is transitively dependent on K , contradiction.

But if X is a proper subset of a key, $X \subset K'$, then A is partially dependent on K' , contradiction.

EN3NF.1 \Rightarrow EN3NF.2

Assume that every non-prime attribute is fully and non-transitively dependent on every key.

And assume that $X \longrightarrow A$ and $A \notin X$, and A is non-prime. We now need to prove X is a superkey.

Every key K determines everything, including X ,
Therefore $K \longrightarrow X \longrightarrow A$.

X must be a key, or a proper subset of a key, otherwise A is transitively dependent on K , contradiction.

But if X is a proper subset of a key, $X \subset K'$, then A is partially dependent on K' , contradiction.

Therefore X is a key, therefore X is a superkey.

EN3NF.2 \Rightarrow *EN3NF.1*

Assume that whenever a non-trivial $X \longrightarrow A$ holds, either X is a superkey or A is prime.

EN3NF.2 \Rightarrow *EN3NF.1*

Assume that whenever a non-trivial $X \longrightarrow A$ holds, either X is a superkey or A is prime.

Let K be a key and A non-prime. We need to show that $K \longrightarrow A$ fully and non-transitively.

EN3NF.2 \Rightarrow *EN3NF.1*

Assume that whenever a non-trivial $X \longrightarrow A$ holds, either X is a superkey or A is prime.

Let K be a key and A non-prime. We need to show that $K \longrightarrow A$ fully and non-transitively.

$K \longrightarrow A$ is non-trivial ($A \notin K$), because because K is a key and A non-prime.

$EN3NF.2 \Rightarrow EN3NF.1$

Assume that whenever a non-trivial $X \longrightarrow A$ holds, either X is a superkey or A is prime.

Let K be a key and A non-prime. We need to show that $K \longrightarrow A$ fully and non-transitively.

$K \longrightarrow A$ is non-trivial ($A \notin K$), because because K is a key and A non-prime.

If $K \longrightarrow A$ is not full, then some $K' \subset K$, ($K' \neq K$) is a non-superkey that determines a non-prime, contradiction.

$EN3NF.2 \Rightarrow EN3NF.1$

Assume that whenever a non-trivial $X \longrightarrow A$ holds, either X is a superkey or A is prime.

Let K be a key and A non-prime. We need to show that $K \longrightarrow A$ fully and non-transitively.

$K \longrightarrow A$ is non-trivial ($A \notin K$), because because K is a key and A non-prime.

If $K \longrightarrow A$ is not full, then some $K' \subset K$, ($K' \neq K$) is a non-superkey that determines a non-prime, contradiction.

If $K \longrightarrow A$ is transitive, then $K \longrightarrow Y \longrightarrow A$ where Y is not a key nor a subset of a key and $A \notin Y$. But non-trivial $Y \longrightarrow A$ gives us that Y is a superkey, contradiction.

$EN3NF.2 \Rightarrow EN3NF.1$

Assume that whenever a non-trivial $X \longrightarrow A$ holds, either X is a superkey or A is prime.

Let K be a key and A non-prime. We need to show that $K \longrightarrow A$ fully and non-transitively.

$K \longrightarrow A$ is non-trivial ($A \notin K$), because because K is a key and A non-prime.

If $K \longrightarrow A$ is not full, then some $K' \subset K$, ($K' \neq K$) is a non-superkey that determines a non-prime, contradiction.

If $K \longrightarrow A$ is transitive, then $K \longrightarrow Y \longrightarrow A$ where Y is not a key nor a subset of a key and $A \notin Y$. But non-trivial $Y \longrightarrow A$ gives us that Y is a superkey, contradiction.

So $K \longrightarrow A$ fully and non-transitively as required.

Boyce-Codd Normal Form (BCNF)

EN3NF whenever a non-trivial $X \longrightarrow A$ holds, either X is a superkey or A is prime

Boyce-Codd Normal Form (BCNF)

EN3NF whenever a non-trivial $X \longrightarrow A$ holds, either X is a superkey or A is prime

BCNF whenever a non-trivial $X \longrightarrow A$ holds, either X is a superkey

That is, it's just the strong form of 3NF, but it also applies to attributes in keys.

On Thursday

We will look at the example in [E&N Figure 10.13]

TEACH(student, course, instructor)

where

$\{student, course\} \longrightarrow instructor$

$instructor \longrightarrow course$

On Thursday

We will look at the example in [E&N Figure 10.13]

TEACH(student, course, instructor)

where

$\{student, course\} \longrightarrow instructor$

$instructor \longrightarrow course$

All attributes are prime, because $\{student, course\}$ and $\{student, instructor\}$ are both candidate keys.

On Thursday

We will look at the example in [E&N Figure 10.13]

$TEACH(student, course, instructor)$

where

$\{student, course\} \longrightarrow instructor$

$instructor \longrightarrow course$

All attributes are prime, because $\{student, course\}$ and $\{student, instructor\}$ are both candidate keys.

So, its in EN3NF, but there seems to be some redundancy here.