COMP 430
Intro. to Database Systems

Normal Forms

Slides use ideas from Chris Ré.

What’s the big idea?

Normal form: Rules for which all schemas must follow.

Normalization: Process for putting schemas in a normal form.

Motivations:

- **Objective** criteria for a **good** design
- A way to fine-tune/fix-up schemas from ER design
- A design technique all by itself – potentially, but not a common view

Common normal forms

- Domain/key normal form

Implies

- 6th normal form

Implies

- 5th normal form

Almost implies

- 4th normal form

Implies

- **Boyce-Codd normal form = 3.5th normal form**

Implies

- **3rd normal form**

Implies

- 2nd normal form

Implies

- 1st normal form

Based on *functional dependencies*, i.e., what attributes depend upon.

Goal: Prevent DB *anomalies*.

Intuition for BCNF & 3NF

(3NF) "[Every] non-key field must provide a fact about the key, the whole key, and nothing but the key." – Bill Kent

(BCNF) "Each attribute must represent a fact about the key, the whole key, and nothing but the key." – Chris Date

Course(<u>crn</u>, dept_code, course_number, title)

Student(<u>student_id</u>, first_name, last_name)

Enrollment(<u>crn</u>, <u>student_id</u>, grade)

Denormalization

We’ll later see that there are also reasons to <u>not</u> use 3NF / BCNF.

1st normal form

instructor	days_teaches
John	{M,T,W,Th,F}
John	{M,T,W,Th,F}
Scott	{T,Th}
…	…

instructor	day_teaches
John	M
John	T
John	W
John	Th
John	F
Scott	T
Scott	Th
…	…

Table represents a mathematical relation:

- Each record unique.
- Single value per record & attribute.

Anomalies

Update, delete, insert

Update anomalies caused by redundancy

student	course	room
John	COMP 130	Lovett Col Com
Jane	COMP 130	Lovett Col Com
John	COMP 430	Keck 100
Mary	COMP 430	Keck 100
Sue	COMP 430	Keck 100
…	…	…

Updating one room results in inconsistency = *update anomaly*.

Assume no separate **Student** or **Course** tables.

Delete anomalies caused by poor attribute grouping

student	course	room
…	…	…

No enrolled students results in no information about course = *delete anomaly*.

Assume no separate **Student** or **Course** tables.

Insert anomalies caused by poor attribute grouping

student	course	room
John	COMP 130	Lovett Col Com
Jane	COMP 130	Lovett Col Com
John	COMP 430	Keck 100
Mary	COMP 430	Keck 100
Sue	COMP 430	Keck 100
…	…	…

Need an enrolled student to reserve a room = *insert anomaly*.

Assume no separate **Student** or **Course** tables.

Functional dependencies

FDs identify relationships between attributes

Process:

1. Start with some relational schema.
2. Identify its functional dependences.
3. Use these to design a better schema.

Not only identify problems, but fix them.

Roadmap

- Define FDs.
- Define closures to find all FDs.
- Define superkeys to determine what should be key.
- Apply definitions to BCNF & 3NF.
- Glimpse at what's beyond BCNF.

With examples & activities, of course!

<u>Intuition:</u>

Attribute X (room) depends on Y (course).

→

X (room) should be a non-key attribute in a table with Y (course) as key.

FD – definition

Let S, T be sets of attributes. Let R be a relation.

S *functionally determines* T ($S \rightarrow T$) holds on R

iff

for all valid tuples t_1, t_2 in R, if $t_1[S]=t_2[S]$, then $t_1[T]=t_2[T]$.

I.e., all the tuples that fit the intended meaning of R.

$S \rightarrow T$ is a *functional dependency*.

How FDs are obtained & used

F = set of FDs
R = relation

- Primarily interested in FDs that hold on all instances (data sets)
  - But can't simply enumerate all instances.
  - Such FDs must come from requirements analysis.
- F can specify a set of constraints on R's tuples.
  - Considered part of R's schema.

Given R, what FDs hold?

Given R, what are its keys?

- To test whether R's tuples are valid under F.

Does R satisfy F?

- To generate more FDs.

Given F, what other FDs hold?

FD – an illustration

		S_1	…	S_m		T_1	…	T_n	
t_1		John	Smith	Jr.		dog	blue	17	
t_2		John	Smith	Jr.		dog	blue	17	

If t_1,t_2 agree here… …they also agree here!

FD example

emp_id	name	phone		position
E0045	Smith	1234	$\leftarrow$	Clerk
E3542	Mike	9876	$\leftarrow$	Salesrep
E1111	Smith	9876	$\leftarrow$	Salesrep
E9999	Mary	1234	$\leftarrow$	Lawyer

{position} $\rightarrow$ {phone}

emp_id	name	phone		position
E0045	Smith	1234	$\rightarrow$	Clerk
E3542	Mike	9876		Salesrep
E1111	Smith	9876		Salesrep
E9999	Mary	1234	$\rightarrow$	Lawyer

Not: {phone} $\rightarrow$ {position}

Activity – Find FDs in this instance

a	b	c	d	e
1	2	4	3	6
3	2	5	1	8
1	4	4	5	7
1	2	4	3	6
3	2	5	1	8

Find at least *three* FDs which hold on this instance:

- $\{\quad\} \rightarrow \{\quad\}$
- $\{\quad\} \rightarrow \{\quad\}$
- $\{\quad\} \rightarrow \{\quad\}$

Where we're headed: defining Superkey

A set S of attributes is a *superkey* of relation R

iff

$S \rightarrow R$,

Here, R means the set of all of the relation's attributes.

i.e., for all valid tuples t_1, t_2 in R, if $t_1[S]=t_2[S]$, then $t_1[R]=t_2[R]$.

Intuition: R's attributes should be dependent on exactly the primary key.

FD implication & closures

Logical implication of FDs – example

name	color	category	dept	price
Gizmo	Green	Gadget	Toys	49
Widget	Black	Gadget	Toys	59
Gizmo	Green	Whatsit	Garden	99

Given:

{name} $\rightarrow$ {color}

{category} $\rightarrow$ {dept}

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

Logically implies:

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

Logical implication rules

Armstrong's Axioms

- Reflexivity: If $B \subseteq A$, then $A \to B$.
- Augmentation: If $A \to B$, then $AC \to BC$.
- Transitivity: If $A \to B$ and $B \to C$, then $A \to C$.

Sound: Only imply correct FDs.

Complete: Imply all correct FDs.

Derivable rules:

- Union: If $A \to B$ and $A \to C$, then $A \to BC$.
- Decomposition: If $A \to BC$, then $A \to B$ and $A \to C$.
- Pseudo-transitivity: If $A \to B$ and $BC \to D$, then $AC \to D$.

Standard notation for union of FD sets.

Activity – using FD implications

name	color	category	dept	price
Gizmo	Green	Gadget	Toys	49
Widget	Black	Gadget	Toys	59
Gizmo	Green	Whatsit	Garden	99

Reflexivity:	If $B \subseteq A$, then $A \rightarrow B$.
Augmentation:	If $A \rightarrow B$, then $AC \rightarrow BC$.
Transitivity:	If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$
Union:	If $A \rightarrow B$ and $A \rightarrow C$, then $A \rightarrow BC$.
Decomposition:	If $A \rightarrow BC$, then $A \rightarrow B$ and $A \rightarrow C$.
Pseudo-transitivity:	If $A \rightarrow B$ and $BC \rightarrow D$, then $AC \rightarrow D$.

Given:

{name} $\rightarrow$ {color}

{category} $\rightarrow$ {dept}

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

Logical implication:

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

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

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

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

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

Rule(s)?

Closures – **all** logically implied FDs

Let F be a set of FDs. Let A, B be sets of attributes.

The *closure* of A (A^+) is the set of all attributes B such that $A \rightarrow B$.

The *closure* of F (F^+) is the set of all FDs logically implied by F.

The *closure* of A (A^+) is the set of all attributes B such that $A \rightarrow B$.

Closure of attribute sets

Given:

{name} $\rightarrow$ {color}

{category} $\rightarrow$ {dept}

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

Some closures:

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

The *closure* of A (A^+) is the set of all attributes B such that $A \to B$.

Algorithm for closure of attribute sets

Closure(A) =

Result = A

Repeat until Result doesn't change:

For each FD $B \to C$:

if $B \subseteq$ Result,

then add C to Result

Return Result

Simple algorithm quadratic in size of F.
Complicated linear algorithm exists.

Given:

$\{name\} \to \{color\}$

$\{category\} \to \{dept\}$

$\{color, category\} \to \{price\}$

Compute:

$\{name, category\}^+$

The *closure* of A (A^+) is the set of all attributes B such that $A \rightarrow B$.

Activity – use closure algorithm

Closure(A) =

Result = A

Repeat until Result doesn't change:

For each FD $B \rightarrow C$:

if $B \subseteq$ Result,

then add C to Result

Return Result

Given:

$\{a, b\} \rightarrow \{c\}$

$\{a, d\} \rightarrow \{e\}$

$\{b\} \rightarrow \{d\}$

$\{a, f\} \rightarrow \{b\}$

Compute:

$\{a, b\}^+ =$

$\{a, f\}^+ =$

The *closure* of F (F^+) is the set of all FDs logically implied by F.

Closure of FD set

Let A, B be sets of attributes.

Closure(F) =

Result = $\varnothing$

For each subset A:

~~For each $A \rightarrow B$ provable~~: For each $B \subseteq A^+$:

Add $A \rightarrow B$ to Result.

Return Result.

The *closure* of F (F^+) is the set of all FDs logically implied by F.

Closure of FD set – example

Given:

$\{a, b\} \rightarrow \{c\}$

$\{a, d\} \rightarrow \{b\}$

$\{b\} \rightarrow \{d\}$

Compute attribute set closures:

$\{a\}^+ = \{a\}$

$\{b\}^+ = \{b, d\}$

$\{c\}^+ = \{c\}$

$\{d\}^+ = \{d\}$

$\{a, b\}^+ = \{a, b, c, d\}$

$\{a, c\}^+ = \{a, c\}$

$\{a, d\}^+ = \{a, b, c, d\}$

$\{b, c\}^+ = \{b, c, d\}$

$\{b, d\}^+ = \{b, d\}$

$\{c, d\}^+ = \{c, d\}$

$\{a, b, c\}^+ = \{a, b, c, d\}$

$\{a, b, d\}^+ = \{a, b, c, d\}$

$\{a, c, d\}^+ = \{a, b, c, d\}$

$\{b, c, d\}^+ = \{b, c, d\}$

$\{a, b, c, d\}^+ = \{a, b, c, d\}$

Compute FDs:

$\{a\} \rightarrow \{a\}$

$\{b\} \rightarrow \{b\}, \ldots \rightarrow \{d\}, \ldots \rightarrow \{b, d\}$

$\{c\} \rightarrow \{c\}$

$\{d\} \rightarrow \{d\}$

$\{a, b\} \rightarrow \{a, b, c, d\}$

$\{a, c\} \rightarrow \{a, c\}$

$\{a, d\} \rightarrow \{a, b, c, d\}$

$\{b, c\} \rightarrow \{b, c, d\}$

$\{b, d\} \rightarrow \{b, d\}$

$\{c, d\} \rightarrow \{c, d\}$

$\{a, b, c\} \rightarrow \{a, b, c, d\}$

$\{a, b, d\} \rightarrow \{a, b, c, d\}$

$\{a, c, d\} \rightarrow \{a, b, c, d\}$

$\{b, c, d\} \rightarrow \{b, c, d\}$

$\{a, b, c, d\} \rightarrow \{a, b, c, d\}$

If we take $A \rightarrow B$ to be shorthand for $A \rightarrow B'$, where $B' \subseteq B$.

The *closure* of F (F^+) is the set of all FDs logically implied by F.

Closure of FD set – example

Given:	**…**	**Compute FDs:**	**Shorthand version:**
$\{a, b\} \rightarrow \{c\}$		$\{a\} \rightarrow \{a\}$	
$\{a, d\} \rightarrow \{b\}$		$\{b\} \rightarrow \{b\}, \ldots \rightarrow \{d\}, \ldots \rightarrow \{b, d\}$	$\{b\} \rightarrow \{d\}$
$\{b\} \rightarrow \{d\}$		$\{c\} \rightarrow \{c\}$	
		$\{d\} \rightarrow \{d\}$	
		$\{a, b\} \rightarrow \{a, b, c, d\}$	$\{a, b\} \rightarrow \{c, d\}$
		$\{a, c\} \rightarrow \{a, c\}$	
		$\{a, d\} \rightarrow \{a, b, c, d\}$	$\{a, d\} \rightarrow \{b, c\}$
		$\{b, c\} \rightarrow \{b, c, d\}$	$\{b, c\} \rightarrow \{d\}$
		$\{b, d\} \rightarrow \{b, d\}$	
		$\{c, d\} \rightarrow \{c, d\}$	
		$\{a, b, c\} \rightarrow \{a, b, c, d\}$	$\{a, b, c\} \rightarrow \{d\}$
		$\{a, b, d\} \rightarrow \{a, b, c, d\}$	$\{a, b, d\} \rightarrow \{c\}$
		$\{a, c, d\} \rightarrow \{a, b, c, d\}$	$\{a, c, d\} \rightarrow \{b\}$
		$\{b, c, d\} \rightarrow \{b, c, d\}$	
		$\{a, b, c, d\} \rightarrow \{a, b, c, d\}$	

Eliminating *trivial* FDs $A \rightarrow B$, where $B \subseteq A$.

Replacing FDs $A \rightarrow AB$ with $A \rightarrow B$.

FD covers & equivalence

F *covers* G iff G can be inferred from F. I.e., $G^+ \subseteq F^+$.
F and G are *equivalent* iff $F^+ = G^+$.

Given:

$\{a, b\} \to \{c\}$
$\{a, d\} \to \{b\}$
$\{b\} \to \{d\}$

closure

Compute FDs:

$\{a\} \to \{a\}$
$\{b\} \to \{b, d\}$
$\{c\} \to \{c\}$
$\{d\} \to \{d\}$
$\{a, b\} \to \{a, b, c, d\}$
$\{a, c\} \to \{a, c\}$
$\{a, d\} \to \{a, b, c, d\}$
$\{b, c\} \to \{b, c, d\}$
$\{b, d\} \to \{b, d\}$
$\{c, d\} \to \{c, d\}$
$\{a, b, c\} \to \{a, b, c, d\}$
$\{a, b, d\} \to \{a, b, c, d\}$
$\{a, c, d\} \to \{a, b, c, d\}$
$\{b, c, d\} \to \{b, c, d\}$
$\{a, b, c, d\} \to \{a, b, c, d\}$

covered by

Shorthand version:

$\{b\} \to \{d\}$
$\{a, b\} \to \{c, d\}$
$\{a, d\} \to \{b, c\}$
$\{b, c\} \to \{d\}$
$\{a, b, c\} \to \{d\}$
$\{a, b, d\} \to \{c\}$
$\{a, c, d\} \to \{b\}$

Superkeys & keys

Superkeys & keys (reminder)

A set S of attributes is a *superkey* of relation R iff S $\rightarrow$ R.

Equivalently, iff $S^+ = R$.

A *key* is a minimal superkey.

We pick a key as *primary key*.

S is a *superkey* of R iff $S \rightarrow R$ iff $S^+ = R$.

Superkeys & keys – example

Product(name, price, category, color)

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

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

What are superkey(s)? Key(s)?

How can you search for them?

Activity

07a-keys.ipynb

“Toy” exercises – Not practical, but help you understand the math.

Normal forms

FD-based table normalization

While there are “bad” FDs

Decompose a table with “bad” FDs into sub-tables.

Boyce-Codd normal form (BCNF)

(BCNF) "Each attribute must represent a fact about the key, the whole key, and nothing but the key." – Chris Date

Ignoring trivial FDs:

- Good: $X \rightarrow R$ X is a (super)key
- Bad: $X \rightarrow A$ for $A \subset R$ X is not a (super)key

"Bad" because X isn't the primary key, but it functionally determines some of the attributes. Thus, there is redundancy.

BCNF – definition

Relation R is in BCNF

iff

whenever A $\rightarrow$ B is a non-trivial FD in R, then A is a superkey for R.

I.e., each FD is trivial or “good”, not “bad”.

BCNF – example

Relation R is in BCNF
iff
whenever $A \rightarrow B$ is a non-trivial FD in R, then A is a superkey for R.

name	ssn	phone	city
Mary	123-45-6789	713-555-1234	Houston
Mary	123-45-6789	713-555-6543	Houston
Joe	987-65-4321	512-555-2121	Austin
Joe	987-65-4321	512-555-1234	Austin

Find a “bad” FD.

$\{ssn\} \rightarrow \{name, city\}$

BCNF – example fixed

Relation R is in BCNF
iff
whenever A → B is a non-trivial FD in R, then A is a superkey for R.

Decompose table into sub-tables.

name	ssn	city
Mary	123-45-6789	Houston
Joe	987-65-4321	Austin

ssn	phone
123-45-6789	713-555-1234
123-45-6789	713-555-6543
987-65-4321	512-555-2121
987-65-4321	512-555-1234

Now a “good” FD.

{ssn} → {name, city}

BCNF decomposition – algorithm

BCNF_decomp(R) =

Find attribute set X s.t. $X^+ \neq X$ (not trivial) and $X^+ \neq R$ (not superkey).

If no such X, then return R.

Let $D = X^+ - X$. (attributes functionally determined by X)

Let $N = R - X^+$. (attributes not functionally determined by X)

Decompose R into $R_1(X \cup D)$ and $R_2(X \cup N)$.

Return BCNF_decomp(R_1), BCNF_decomp(R_2).

BCNF decomposition – example

BCNF_decomp(R) =

Find attribute set X s.t. $X^+ \neq X$ and $X^+ \neq R$.

If no such X, then return R.

Let $D = X^+ - X$.

Let $N = R - X^+$.

Decompose R into $R_1(X \cup D)$ and $R_2(X \cup N)$.

Return BCNF_decomp(R_1), BCNF_decomp(R_2).

$R(a,b,c,d,e)$
$\{a\} \rightarrow \{b, c\}$
$\{c\} \rightarrow \{d\}$

$X = \{a\}$
$X^+ = \{a,b,c,d\}$

$D = \{b,c,d\}$
$N = \{e\}$

$R_1(a,b,c,d)$
$R_2(a,e)$

BCNF decomposition – example

BCNF_decomp(R) =

Find attribute set X s.t. $X^+ \neq X$ and $X^+ \neq R$.

If no such X, then return R.

Let $D = X^+ - X$.

Let $N = R - X^+$.

Decompose R into $R_1(X \cup D)$ and $R_2(X \cup N)$.

Return BCNF_decomp(R_1), BCNF_decomp(R_2).

$R_1(a,b,c,d)$
$\{a\} \rightarrow \{b, c\}$
$\{c\} \rightarrow \{d\}$

$X = \{c\}$
$X^+ = \{c,d\}$

$D = \{d\}$
$N = \{a,b\}$

$R_{11}(c,d)$
$R_{12}(a,b,c)$

BCNF decomposition – example

$R_{11}(c,d)$
$\{a\} \rightarrow \{b, c\}$
$\{c\} \rightarrow \{d\}$

No such X.

BCNF_decomp(R) =

Find attribute set X s.t. $X^+ \neq X$ and $X^+ \neq R$.

If no such X, then return R.

Let $D = X^+ - X$.

Let $N = R - X^+$.

Decompose R into $R_1(X \cup D)$ and $R_2(X \cup N)$.

Return BCNF_decomp(R_1), BCNF_decomp(R_2).

BCNF decomposition – example

$R_{12}(a,b,c)$
$\{a\} \to \{b, c\}$
$\{c\} \to \{d\}$

BCNF_decomp(R) =

Find attribute set X s.t. $X^+ \neq X$ and $X^+ \neq R$.

If no such X, then return R.

Let $D = X^+ - X$.

Let $N = R - X^+$.

Decompose R into $R_1(X \cup D)$ and $R_2(X \cup N)$.

Return BCNF_decomp(R_1), BCNF_decomp(R_2).

No such X.

BCNF decomposition – example

$R_2(a,e)$
$\{a\} \rightarrow \{b, c\}$
$\{c\} \rightarrow \{d\}$

BCNF_decomp(R) =

Find attribute set X s.t. $X^+ \neq X$ and $X^+ \neq R$.

If no such X, then return R.

No such X.

Let $D = X^+ - X$.

Let $N = R - X^+$.

Decompose R into $R_1(X \cup D)$ and $R_2(X \cup N)$.

Return BCNF_decomp(R_1), BCNF_decomp(R_2).

BCNF decomposition + keys – example

$R(a,b,c,d,e)$
$\{a\} \to \{b, c\}$
$\{c\} \to \{d\}$

BCNF

$R_{11}(c,d)$
$R_{12}(a,b,c)$
$R_2(a,e)$
$\{a\} \to \{b, c\}$
$\{c\} \to \{d\}$

keys

$R_{11}(\underline{c},d)$
$R_{12}(\underline{a},b,c)$
$R_2(\underline{a},\underline{e})$
$\{a\} \to \{b, c\}$
$\{c\} \to \{d\}$

Activity – BCNF & keys

BCNF_decomp(R) =
Find attribute set X s.t. $X^+ \neq X$ and $X^+ \neq R$.
If no such X, then return R.

Let D = X^+ - X.
Let N = R - X^+.
Decompose R into $R_1(X \cup D)$ and $R_2(X \cup N)$.
Return BCNF_decomp(R_1), BCNF_decomp(R_2).

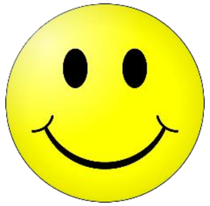
07b-bcnf.ipynb

name	ssn	phone	city	zip
Mary	123-45-6789	713-555-1234	Houston	77005
Mary	123-45-6789	713-555-6543	Houston	77005
Joe	987-65-4321	281-555-2121	Houston	77005
Joe	987-65-4321	281-555-1234	Houston	77005

$\{city\} \rightarrow \{zip\}$
$\{ssn\} \rightarrow \{name, city\}$

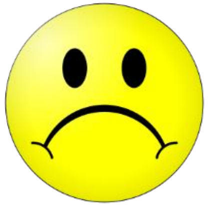
Decompositions

BCNF summary so far – pros/cons



Algorithm to detect & remove redundancies.

Standard practice.



Sometimes some subtle, undesirable side-effects.

Decompositions & joins

$R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p)$

Decomp.

Join???

$R_1(A_1, \ldots, A_n, B_1, \ldots, B_m)$
$R_2(A_1, \ldots, A_n, C_1, \ldots, C_p)$

If decomposition is *lossless*, a join restores the original relation.

Decomposition & join – lossless example

name	price	category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Decomp.

Join

name	price
Gizmo	19.99
OneClick	24.99
Gizmo	19.99

name	category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Decomposition & join – lossy example

name	price	category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Decomp.

price	category
19.99	Gadget
24.99	Camera
19.99	Camera

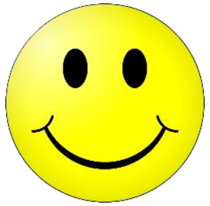
name	category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Join

name	price	category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	24.99	Camera
OneClick	19.99	Camera
Gizmo	19.99	Camera

Loses the association between **name** and **price**.

BCNF decomposition is lossless



$R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p)$

Decomp.

Join

Lossless iff

$\{A_1, \ldots, A_n\} \rightarrow \{B_1, \ldots, B_m\}$

$R_1(A_1, \ldots, A_n, B_1, \ldots, B_m)$

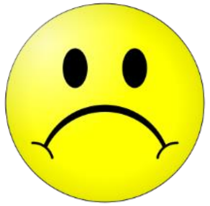
$R_2(A_1, \ldots, A_n, C_1, \ldots, C_p)$

Don't need

$\{A_1, \ldots, A_n\} \rightarrow \{C_1, \ldots, C_p\}$

Holds by definition of BCNF decomposition algorithm.

BCNF can lose FD information



R(name, company, category)
{category, company} $\to$ {name}
{name} $\to$ {company}

Decomp.

R_1(<u>name</u>, <u>category</u>)
R_2(<u>name</u>, company)
{category, company} $\to$ {name}
{name} $\to$ {company}

Keys: {category, company}, {category, name}

"Bad" FD

Can't enforce "nonlocal" FD.

<u>name</u>	<u>category</u>
Gizmo	Gadget
GizmoPlus	Gadget

<u>name</u>	company
Gizmo	GizmoWorks
GizmoPlus	GizmoWorks

Join

name	company	category
Gizmo	GizmoWorks	Gadget
GizmoPlus	GizmoWorks	Gadget

Three solutions

- Accept the BCNF tradeoff between avoiding redundancy/anomalies and preserving FDs.

  BCNF is most common choice.

- Take extra steps to enforce these FDs.

  E.g., join tables and then check.

- Weaken decomposition so that no such lost FDs.

  E.g., 3NF.

3^{rd} normal form (3NF)

BCNF:

Relation R is in BCNF

iff

whenever $A \rightarrow B$ is a non-trivial FD in R, then

- A is a superkey for R.

3NF:

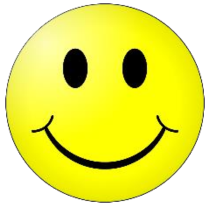
Relation R is in 3NF

Iff

whenever $A \rightarrow B$ is a non-trivial FD in R, then either:

- A is a superkey for R, or
- Every element of B is part of a key.

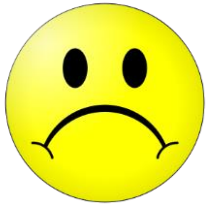
3NF avoids losing FD information



R(name, company, category)
{category, company} $\rightarrow$ {name}
{name} $\rightarrow$ {company}

BCNF: “Bad”
3NF: “Good” because **company** part of a key.

Keys: {category, company}, {category, name}

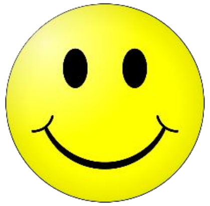
3NF allows some redundancies/anomalies 

name	company	category
Gizmo	GizmoWorks	Gadget
Gizmo	GizmoWorks	Camera
GizmoPlus	GizmoWorks	Gadget
GizmoPlus	GizmoWorks	Camera
NewThing	NULL	Gadget

Redundancy. Repeating product's company.

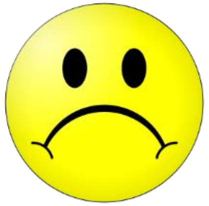
Insertion anomaly. Product not yet made by any company.

3NF summary – pros/cons



3NF is still lossless!

Another common standard.



Allows some redundancies/anomalies.

Requires somewhat more complicated decomposition algorithm.

Glimpse beyond BCNF/3NF/FDs

- 4NF – multi-valued dependencies (generalizes FDs)
- 5NF & 6NF – join dependencies
- DKNF – only domain & key constraints

Multi-value dependencies

FD: $A \rightarrow B$ The value in A determines a value for B.

MVD: $A \twoheadrightarrow B$ The value in A determines a set of values for B.

restaurant	menu_item	delivery_area
Papa John's	Pizza	Rice Village
Papa John's	Pizza	Rice University
Papa John's	Pizza	Southampton
Domino's	Pizza	Rice Village
Domino's	Pizza	Rice University
Domino's	Pasta	Rice Village
Domino's	Pasta	Rice University

$\{restaurant\} \twoheadrightarrow \{menu\_item\}$

$\{restaurant\} \twoheadrightarrow \{delivery\_area\}$

Normal form summary

- Constraints on data/tables to limit redundancy
- Decomposition strategy/algorithm to meet constraints
- Different normal forms for different trade-offs