

Normalization

Thanks to Lois Delcambre

© Lois Delcambre, David Maier 2005-2016

1

Normalization based on Functional Dependencies (FDs)

- **Normalization** works on relations in a relational database.
- **Normalization** based on FDs has been formalized and the theory has been completely worked out.
- **Normalization** makes it easier to update a database but may degrade query performance – an engineering tradeoff.

© Lois Delcambre, David Maier 2005-2016

2

Normalization Strand Map

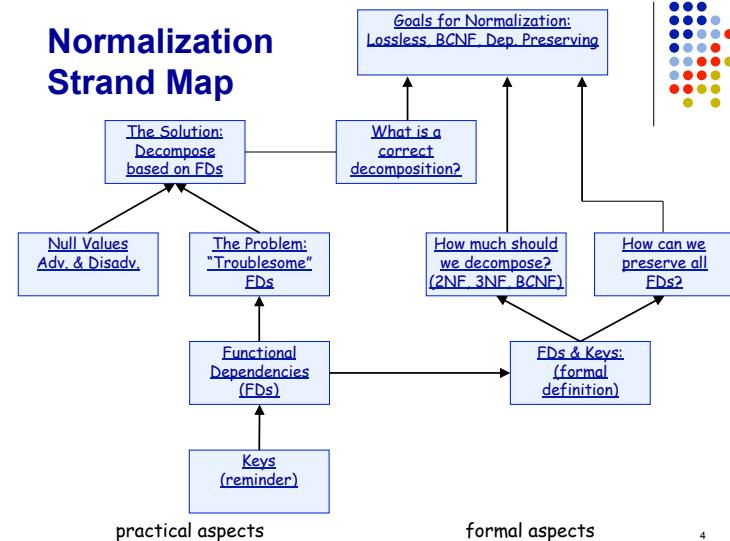
*All computer science students must learn to integrate **theory** and **practice**, to recognize the importance of **abstraction**, and to appreciate the value of **good engineering design**.* CC2001 (<http://www.sigcse.org/cc2001/>)

- This normalization strand map has two strands:
 - the practical concepts and techniques and
 - the formal concepts and results for normalization.

© Lois Delcambre, David Maier 2005-2016

3

Normalization Strand Map



Keys for a Table (reminder)

The key(s) for a table must have unique values and the key(s) for a table help us understand what the table is “about.”

© Lois Delcambre, David Maier 2005-2016

5

Notice ... only one value for non-key attributes (for each key value)

Employee	ssn	name	salary	job-code
	111111111	John Smith	40,000	15
	123456789	Mary Smith	50,000	22
	123456789	Maria Jones	50,000	24

1. NOT allowed because ssn is key!

2. Only one name (and one salary and one job-code) for each row.

For one particular ssn value, **123-45-6789**, there is only **ONE** name because

1. there is only one tuple and
2. we assume that attributes values are atomic.

© Lois Delcambre, David Maier 2005-2016

6

Functional Dependencies (FDs) generalize keys

Functional dependencies (FDs) for relational tables are a generalization of the notion of key for a table.

© Lois Delcambre, David Maier 2005-2016

7

Functional Dependencies

An FD, $X \xrightarrow{\text{"determines"}} Y$, where X is a set of attributes and Y is a set of attributes

It is a statement that if two tuples agree on attributes X they must agree on attributes Y

For each X value there is **ONLY** one Y value.

© Lois Delcambre, David Maier 2005-2016

8

Functional Dependencies (from semantics of the application)

Likely **functional dependencies**:

$ssn \rightarrow employee\text{-}name$

$course\text{-}number\ dept \rightarrow course\text{-}title$

Unlikely **functional dependencies**

$dept \rightarrow book$ **X**

$birthdate$ **X** ssn

© Lois Delcambre, David Maier 2005-2016

9

Will FDs be enforced?

Consider this table:

Emp(ssn, name, phone, dnum, dept-name)

Suppose there is an FD from $dnum \rightarrow dept\text{-}name$

But ssn is the key for this table.

What will prevent two names for one dept?

© Lois Delcambre, David Maier 2005-2016

10

Will this FD be enforced? Let's try it.

Consider this table:

Emp(ssn, name, phone, dnum, dept-name)

Employee	ssn	Name	Phone	Dnum	Dept-name
	111111111	John	555-1234	12	Sales
	222222222	Mary	555-7890	12	Marketing
	...				

Can we put these two rows in this table?

Yes, it doesn't violate the key constraint.

But, the FD from dept to dept-name is violated! We shouldn't have two different names for dnum 12!

© Lois Delcambre, David Maier 2005-2016

11

Functional Dependencies

For an FD

$$X \rightarrow Y$$

We say that X *determines* Y

We want to know if it is **always true** in the application.

We can then use FDs to figure out the keys for tables and to normalize the tables.

© Lois Delcambre, David Maier 2005-2016

12

Every key implies a set of FDs



Each key implies a set of functional dependencies (FDs) from the key to the non-key attributes.

Employee (ssn, name, salary, job-code)

FDs implied by the key:

ssn \rightarrow name
ssn \rightarrow salary
ssn \rightarrow job-code

© Lois Delcambre, David Maier 2005-2016

13

But, some FDs are NOT implied by the key.



Emp(ssn, name, phone, dnum, dept-name)

There is an FD from *dnum* \rightarrow *dept-name*

© Lois Delcambre, David Maier 2005-2016

14

The Problem: “Troublesome” FDs

“Troublesome” FDs (FD where the left-hand-side of the FD is NOT a key for the table where its attributes appear) cause redundancy and update anomalies.



© Lois Delcambre, David Maier 2005-2016

15

Advantages & Disadvantages of Redundancy



- **Disadvantage:** Any time information is stored more than once, it has the possibility of being **inconsistent**.
 - Phone numbers in your laptop
 - Phone numbers in your cell phone
 - Phone numbers in your address book

If someone changes his or her phone number, do you remember to change it in every place?

- **Advantage:** Redundant information may improve **retrieval speed**

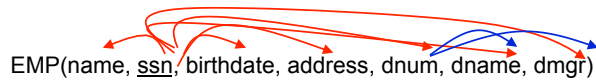
© Lois Delcambre, David Maier 2005-2016

16

Sometimes Redundancy is Caused by FDs



Consider this table:



Then **dname** and **dmgr** are stored redundantly – whenever there are multiple employees in a department.

This redundancy is caused by what we informally call “troublesome” FDs. The FDs shown in blue are “troublesome”.

© Lois Delcambre, David Maier 2005-2016

17

Redundancy Caused by Troublesome FD – Sample Data



EMP(name, ssn, birthdate, address, dnum, dname, dmgr)

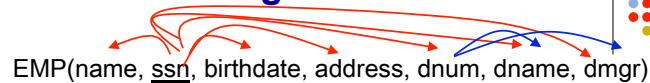
John	111	June 3	123 St.	D1	sales	222
Sue	222	May 15	455 St.	D1	sales	222
Max	333	Mar. 5	678 St.	D2	research	333
Wei	444	May 2	999 St.	D2	research	333
Tom	555	June 22	888 St.	D2	research	333

We have the department name and manager twice for D1 and three times for D2!

© Lois Delcambre, David Maier 2005-2016

18

What's wrong?



The name and the manager of the department is **repeated**, for each employee that works in that department.

Redundancy!

If you replicate information, the copies might be inconsistent.

© Lois Delcambre, David Maier 2005-2016

19

Update Anomalies caused by “troublesome” FDs

EMP(name, ssn, birthdate, address, dnum, dname, dmgr)

Insertion anomalies:

if you insert an employee with a department then you need to know the descriptive information for that department.

if you want to insert a department, you can't ... until there is at least one employee.

Deletion anomalies: if you delete an employee, is that dept. gone? Was this the last employee in that dept.?

Modification anomalies: If you want to change **dname**, for example, you need to change it everywhere! And you have to find them all first.

Troublesome FDs cause (redundancy and) update anomalies.

© Lois Delcambre, David Maier 2005-2016

20

Null Values: Advantages & Disadvantages

Null values are useful
but they can make SQL
queries more complex.



© Lois Delcambre, David Maier 2005-2016

21

Null Values are Useful

Consider

Employee(ssn, name, DOB, *partner* spouse)

Null values make it simpler to insert data, for
example

- before you know the spouse name
- when there is no spouse

Allowing null values makes the DB more flexible.



© Lois Delcambre, David Maier 2005-2016

22

Null Values Cause Problems for Aggregate Operators

Employee (ssn, name, salary) *not allowed*

SELECT AVG(salary) FROM Employee;

SELECT SUM(salary) INTO salsum FROM Employee;
SELECT COUNT(*) INTO total FROM Employee;

salsum/total might be different from first query answer.
How could that happen?

They can also complicate joins and WHERE clauses



© Lois Delcambre, David Maier 2005-2016

23

Splitting Tables Reduces the Need for Null Values

Use two tables:

Employee (ssn, name, DOB)
Employee-extra (ssn, spouse)

Rather than:

Employee(ssn, name, DOB, spouse)

Generally, it is better to reduce the use of null values,
if you can. The first design, above, doesn't require the
use of null values for spouse.



© Lois Delcambre, David Maier 2005-2016

24

The Solution: Lifting “Troublesome” FDs

Normalization by decomposition, based on FDs (where “troublesome” FDs are lifted into a separate table), reduces redundancy and update anomalies.

© Lois Delcambre, David Maier 2005-2016

25

Example: Finding Troublesome FDs

EMP(name, ssn, birthdate, address, dnum, dname, dmgr)

We have a problem!
dnum is NOT the key for this table!

So these blue FDs will not be enforced automatically by the DBMS (using only keys).
And there can be redundancy and update anomalies

© Lois Delcambre, David Maier 2005-2016

26

Example: Lifting Troublesome FDs

EMP(name, ssn, birthdate, address, dnum, dname, dmgr)

1. Lift the “troublesome” FD into its own table with dnum as the key. Now they will be enforced.

Dept(dnum, dname, dmgr)

2. Leave the LHS of the “troublesome” FDs behind. Define a foreign key where New-Emp.dnum REFERENCES Dept.dnum

New-Emp(name, ssn, birthdate, address, dnum)

© Lois Delcambre, David Maier 2005-2016

27

Table is Split onto New Schemas

New-EMP(name, ssn, birthdate, address, dnum)

John	111	June 3	123 St.	D1
Sue	222	May 15	455 St.	D1
Max	333	Mar. 5	678 St.	D2
Wei	444	May 2	999 St.	D2
Tom	555	June 22	888 St.	D2

Dept(dnum, dname, dmgr)

D1	sales	222
D2	research	333

Less redundancy!

© Lois Delcambre, David Maier 2005-2016

28

Basic Idea: Normalize based on FDs



- Identify all the (non-trivial) FDs in an application.
 - Identify FDs that are implied by the keys.
 - Identify FDs that are NOT implied by the keys – the “troublesome” ones.
- Decompose a table with a “troublesome” FD into two or more tables by “lifting” each troublesome FD into a table of its own. Note: when there are two or more “troublesome” FDs with the same left side, then they can be lifted, together, into a single table.

© Lois Delcambre, David Maier 2005-2016

29

Can Define a View to Get Original Table



Emp(name, ssn, birthdate, address, dnum, dname, dmgr)
split into

Dept(dnum, dname, dmgr)

New-Emp(name, ssn, birthdate, address, dnum)

If there are applications that currently query Emp, can define a view:

```
CREATE VIEW Emp AS
SELECT *
FROM Dept NATURAL JOIN New-Emp
```

Update statements will require changes in most cases

© Lois Delcambre, David Maier 2005-2016

30

Advantages of Normalization based on Decomposition



When this table:

Emp(name, ssn, birthdate, address, dnum, dname, dmgr)

is replaced by these two tables:

Dept(dnum, dname, dmgr)

New-Emp(name, ssn, birthdate, address, dnum)

Are there any update anomalies in the new tables?

© Lois Delcambre, David Maier 2005-2016

31

Let's Check the Update Anomalies



Insertion anomalies:

if you insert an employee with a department
then you need to know the descriptive information for
that department. **NO – ONLY THE NUMBER**

if you want to insert a department, you can't ... until
there is at least one employee. **NO PROBLEM**

Deletion anomalies: if you delete an employee, is that dept.
gone? Was this the last employee in that dept.? **NO PROBLEM**

Modification anomalies: If you want to change *dname*, for
example, you need to change it everywhere! And you
have to find them all first. **dname is only stored once!**

Is there any redundancy? **Yes – in the foreign key.**

© Lois Delcambre, David Maier 2005-2016

32

Questions about normalization

- How do we know which FDs we have?
Talk to domain experts; identify FDs; use them as the starting point for normalization. *lossless*
- How do we know if the decomposition is correct?
- How do we know how much to normalize?
How far should we go? *normal forms*
- How do we know if all of the FDs of interest are being enforced – by using keys for tables?
We need the formal definition of FDs to be able to answer these questions.

© Lois Delcambre, David Maier 2005-2016

33

FDs and Keys: Formal Definition

A functional dependency is formally defined as a functional relationship between two sets of attributes. This leads to the definition of trivial FDs and superkeys.

© Lois Delcambre, David Maier 2005-2016

34

Definition of a function

Remember the definition of a **function**:

x	f(x)	x	g(x)	x	h(x)
1	2	1	2	1	10
1	3	2	2	2	20
2	5	3	5	3	30
3	5				

Which of these are functions?

An FD is a **functional** relationship (that **must** occur in a relation) among attribute values

© Lois Delcambre, David Maier 2005-2016

35

Answer (concerning functions)

Remember the definition of a **function**:

x	f(x)	x	g(x)	x	h(x)
1	2	1	2	1	10
1	3	2	2	2	20
2	5	3	5	3	30
3	5				

f is NOT a function because for an input of "1" there are two answers ("2" and "3").
g and h are functions.

© Lois Delcambre, David Maier 2005-2016

36

Example of an FD – a function

Employee (ssn, name, phone, salary)

Since $ssn \rightarrow name$ is an FD

If we know that there is only one name for an ssn,

then we know that $ssn \rightarrow name$ is a **function**

We don't expect $salary \rightarrow phone$ to be function.



© Lois Delcambre, David Maier 2005-2016

37

Another Example

Employee (ssn, name, phone, dept, dept-mgr)

$dept \rightarrow dept-mgr$

If we know that there is only one dept-mgr for a dept,

then we know that $dept \rightarrow dept-mgr$ is a **function!**



© Lois Delcambre, David Maier 2005-2016

38

Trivial FD

We have a trivial FD whenever the attributes on the right side of an FD are a subset of the attributes on the left side of the FD:

$name\ phone \rightarrow phone$

Trivial FDs aren't "troublesome" and won't help us decompose a table.



© Lois Delcambre, David Maier 2005-2016

39

Definition of a Superkey for a Relation

A *superkey* is a set of fields from a relation that contains a key.

Every key is (automatically) a superkey.

A superkey is NOT necessarily a key.

Example:

Emp (ssn, name, phone, dept)

ssn is a key for this relation.

(dept, ssn) is a superkey for this relation.



© Lois Delcambre, David Maier 2005-2016

40

Definition of a Key for a Relation



A key is a **minimal** set of attributes in a relation whose values are guaranteed to uniquely identify tuples in the relation.

- Two distinct tuples have distinct key values
- (minimal) No subset of the fields that comprise a key is a key
- Can be more than one key for a table (not just a single declared key as in SQL)

© Lois Delcambre, David Maier 2005-2016

41

Keys and FDs are Constraints



- We need to know if keys and FDs (always) hold in the application.
- We need to consult a domain expert to find out what the keys and FDs are. The keys and FDs serve as input to the database design process.

© Lois Delcambre, David Maier 2005-2016

42

2NF, 3NF, BCNF: Normal forms based on FDs

Given a set of FDs and one or more tables, three increasingly stronger normal forms, namely 2NF, 3NF, and BCNF, have been defined.

BCNF implies 3NF.

3NF implies 2NF.

(BCNF = Boyce-Codd Normal Form)



© Lois Delcambre, David Maier 2005-2016

43

Informal Definitions Normal Forms Based on FDs



1NF - all attribute values (domain values) are atomic
(part of the definition of the relational model)

2NF - all non-key* attributes must depend on a **whole** key (no partial dependencies)

$r(A \underline{B} C D E) \quad B \rightarrow C$ violates 2NF

3NF - table is in 2NF and all non-key attributes must depend on **only** a key (no transitive dependencies)

$r(A \underline{B} C D E) \quad C \rightarrow D$ violates 3NF

BCNF - every left side of an FD is a key for the table
(All FDs are implied by the keys)

$r(A \underline{B} C D E) \quad C \rightarrow A$ violates BCNF (but not 3NF)

* "non-key" = not part of any key *Two keys* *part of a key (AB)*

© Lois Delcambre, David Maier 2005-2016

44

Examples of Violations

2NF - all **non-key** attributes must depend on a whole key
Assigned-to (a-project, a-emp, emp-name, percent)

3NF - all **non-key** attributes must depend on only a key
Employee (ssn, name, address, project, p-title)

BCNF - every determinant (LHS of an FD) is a key for this table
(all FDs are implied by the keys) $\text{emp-ID} \rightarrow \text{ssn}$

Assigned-to (emp-ID, a-project, ssn, percent)

© Lois Delcambre, David Maier 2005-2016

45

Fix violations of Normal Forms by lifting “troublesome” FDs

Assigned-to (a-project, a-emp, emp-name, percent)

Employee (a-emp, emp-name)

1. Lift the troublesome FD(s) into a table of their own.
Key for new table is left hand side of the troublesome FD.
2. Leave the left side of the FD behind in the original table.
Assigned-to (a-project, a-emp, percent)
3. Eliminate *emp-name* from the *Assigned-to* table.

© Lois Delcambre, David Maier 2005-2016

46

Formal definition of BCNF

For a table R, every FD $X \rightarrow A$ that occurs among attributes of R then either:

- A is an element of X ($X \rightarrow A$ is trivial), or
- X is a superkey of R

For 3NF there is one other option:

- A is part of a key

© Lois Delcambre, David Maier 2005-2016

47

Dependency Preservation: Using a sound & complete set of inference rules

Dependency preservation requires the use of a sound and complete set of rules of inference to compute F^+ , the closure of a set F of FDs. Given the original set of FDs, F. Let G consist of the FDs in F^+ whose attributes appear in any relation scheme (after normalization). Dependency preservation is when $F^+ = G^+$.

© Lois Delcambre, David Maier 2005-2016

48

Example

Consider the following table:

Employee (ssn, name, phone, dept, dept-name)

Original FDs F:

ssn → name ssn → phone ssn → dept
ssn → dept-name dept → dept-name

Employee (ssn, name, phone, dept)

Department (dept, dname)

Resulting FDs G:

ssn → name ssn → phone ssn → dept
dept → dept-name

What about **ssn → dept-name**? Is it lost? Can there be two department names for one ssn?

© Lois Delcambre, David Maier 2005-2016

49

Example (cont.)

Employee (ssn, name, phone, dept, dept-name)

Original FDs F:

ssn → name ssn → phone ssn → dept
ssn → dept-name dept → dept-name

Employee (ssn, name, phone, dept)

Department (dept, dname)

Resulting FDs G:

ssn → name ssn → phone ssn → dept
dept → dept-name

What about **ssn → dept-name**? Is it lost? Can there be two department names for one ssn?

NO! It's not lost. One ssn has only one dept. And one dept has only one dept-name. So ssn has only one dept-name.

© Lois Delcambre, David Maier 2005-2016

50

We need to derive all FDs from a given set of FDs. We need rules.

For sets of attribute X and Y

Reflexivity *All trivial dependencies*

If Y is a subset of X, then $X \rightarrow Y$

As an example, for all attributes, $A \rightarrow A$

examples: name → name, gender → gender

Augmentation

If FD $X \rightarrow Y$ holds, then so does $XZ \rightarrow YZ$, for all Z

As an example, augmentation creates superkeys from keys.

Transitivity

If FDs $X \rightarrow Y$ and $Y \rightarrow Z$ hold,
then so does $X \rightarrow Z$

© Lois Delcambre, David Maier 2005-2016

51

Use the rules to compute closure

Let F be a set of FDs.

F⁺ is the set of all FDs **implied** (or **derivable**) from F using a sound & complete set of inference rules

Reflexivity: If Y is a set of attrs, Y subset of X, then $X \rightarrow Y$

Augmentation: If $X \rightarrow Y$, and Z is a set of attrs, then $XZ \rightarrow XY$

Transitivity: If $X \rightarrow Y$, $Y \rightarrow Z$, then $X \rightarrow Z$

Compute F⁺ by applying rules until no new FDs arise.

F⁺ is called the **closure** of F)

© Lois Delcambre, David Maier 2005-2016

52

Definition of Dependency Preserving



Suppose F is the original set of FDs.

Compute F^+ .

G is set of FDs from F^+ that are present in individual relations in G . *after decomposition*

Compute G^+ .

If $F^+ = G^+$

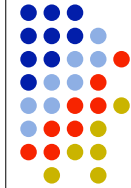
then the decomposition is **dependency preserving**

For a complex design, you may want to implement one of the known algorithms for testing $F^+ = G^+$.

© Lois Delcambre, David Maier 2005-2016

53

Decomposition is correct when it is lossless



The decomposition algorithm (based on lifting “troublesome” FDs into a separate table) guarantees that the decomposition of the original table is **lossless**.

© Lois Delcambre, David Maier 2005-2016

54

Decompose: Project Operator Recompose: Join Operator



When

$Emp(name, \underline{ssn}, birthdate, address, dnum, dname, dmgr)$

is replaced by these two tables:

$Department(dnum, dname, dmgr)$

$NewEmp(name, \underline{ssn}, birthdate, address, dnum)$

We use the project operator to decompose

$Department = \pi_{dnum, dname, dmgr} Emp$

$NewEmp = \pi_{name, ssn, birthdate, address, dnum} Emp$

And we use the join operator to put the pieces together

$Emp = Department \bowtie_{D.dnum=NE.dnum} NewEmp$

© Lois Delcambre, David Maier 2005-2016

55

What is a lossless (and a lossy) decomposition?



We want to make sure that we haven't thrown away any information from the original schema.

When table R is decomposed into tables R_1 and R_2 then the decomposition is **lossless (correct)** if:

$(R_1 \bowtie R_2)$ is identical to R *natural join*

If it is a **lossy** decomposition, then $R_1 \bowtie R_2$ gives you **TOO MANY** tuples.

© Lois Delcambre, David Maier 2005-2016

56

Example: a lossy decomposition

original

Employee	(emp-number, name, p-num, p-title)
1	smith p1 accounting
2	jones p1 accounting
3	smith p2 billing

decomposition:

Employee	(emp-number, name)
1	smith
2	jones
3	smith

Project	(p-num, p-title, name)
p1 account	smith
p1 account	jones
p2 billing	smith

now with natural join: you get at least one extra tuple!

$\frac{1}{3}$ smith p2 billing
smith p1 account

© Lois Delcambre, David Maier 2005-2016

57

One Guarantee for a Lossless Decomposition

Consider a table:

$R(\underline{a}, b, c, d, e)$ with a troublesome FD $d \rightarrow e$.

Decompose it into two tables:

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

$R_2(\underline{d}, e)$

As long as

the attributes in common are a key for (at least) one of the relations, R_1 or R_2

then we know that the decomposition is lossless.

For this example d is the attribute in common.

And d is a key for R_2 , the second table.

© Lois Delcambre, David Maier 2005-2016

58

Is the Decomposition Algorithm Lossless?

1. Lift the "troublesome" FD(s) (all the FDs with the same LHS) into a table of their own. Key for new table is left hand side of the troublesome FD(s).

2. Leave the left side of the FD behind in the original table.

3. Eliminate the RHS attributes from the original table.

1

Yes, we are guaranteed that the decomposition is lossless. The attribute in common is definitely a key for the new "lifted" table.

© Lois Delcambre, David Maier 2005-2016

59

Example of a Lossy Decomposition (revisited)

Employee(emp-number, name, p-num, p-title)

decomposition: Employee (emp-number, name)

Project (p-num, p-title, name)

together form the key

Notice that the common attribute, name, is not a key for either of these tables.

© Lois Delcambre, David Maier 2005-2016

60

Three Goals for Normalization

lossless decomposition
 don't throw any information away
 be able to reconstruct the original relation

dependency preservation
 all of the original, non-trivial FDs can be derived
 from FDs implied by the keys of resulting tables

Boyce-Codd normal form (BCNF) - no redundancy
 beyond foreign keys; all FDs implied by keys

© Lois Delcambre, David Maier 2005-2016

61

It is not always possible to have BCNF AND dependency preservation

Required! → **lossless** decomposition

Desirable but not always possible to have both

- **dependency preservation**
- **Boyce-Codd normal form (BCNF)**

© Lois Delcambre, David Maier 2005-2016

62

Counterexample (a table that can't be decomposed into BCNF with dependency preservation)

Original table – a table that holds US addresses

addr(number street city state zip)

The original FDs are:

number street city state → *zip*
zip → *state*

© Lois Delcambre, David Maier 2005-2016

63

Counterexample (cont.)

Based on the FDs:

number street city state → *zip*
zip → *state*

There are two keys for this table

addr(number street city state zip)

Since all attributes are key attributes, this table is automatically in 3NF and 2NF.

But *zip* → *state* violates BCNF

© Lois Delcambre, David Maier 2005-2016

64

Counterexample (cont.)

Let's decompose

addr(number street city state zip)

using this "troublesome" FD:

$zip \rightarrow state$

Addr2 (number, street, city, zip)

Zip-state (zip, state)

$ns cz \rightarrow nscz$
 $z \rightarrow s$

We've lost the FD $number\ street\ city\ state \rightarrow zip$

If we put this table back in the design, we are back where we started. And we violate BCNF.

© Lois Delcambre, David Maier 2005-2016

65

Algorithm for lossless join decomposition into BCNF relations

(not necessarily dependency preserving)

1. set $D := \{ R \}$ (the current set of relations)

2. while there is a relation in R that is not in BCNF
begin

choose a relation Q that is not in BCNF

find an FD $X \rightarrow Y$ in Q that violates BCNF

replace Q in D by two relations: $(Q - Y)$ and $(X \cup Y)$

end;

Lifting "troublesome" FD

Finding a "troublesome" FD

© Lois Delcambre, David Maier 2005-2016

66