

Carnegie Mellon Univ.
Dept. of Computer Science
15-415 - Database Applications


Lecture #17: Schema Refinement &
Normalization - Normal Forms
(R&G, ch. 19)



Overview - detailed

- DB design and normalization
 - pitfalls of bad design
 - decomposition
 - normal forms

Faloutsos CMU SCS 15-415 2



Goal

- Design ‘good’ tables
 - sub-goal#1: define what ‘good’ means
 - sub-goal#2: fix ‘bad’ tables
- in short: “*we want tables where the attributes depend on the primary key, on the whole key, and nothing but the key*”
- Let’s see why, and how:

Faloutsos CMU SCS 15-415 3

CMU SCS

Pitfalls

takes1 (ssn, c-id, grade, name, address)

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main

Faloutsos CMU SCS 15-415 4

CMU SCS

Pitfalls

'Bad' - why? because: ssn->address, name

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main

Faloutsos CMU SCS 15-415 5

CMU SCS

Pitfalls

- Redundancy
 - space
 - (inconsistencies)
 - insertion/deletion anomalies:

Faloutsos CMU SCS 15-415 6

Pitfalls

- insertion anomaly:
 - “jones” registers, but takes no class - no place to store his address!

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
...
234	null	null	jones	Forbes

Faloutsos CMU SCS 15-415 7

Pitfalls

- deletion anomaly:
 - delete the last record of ‘smith’ (we lose his address!)

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main


Faloutsos CMU SCS 15-415 8

Solution: decomposition

- split offending table in two (or more), eg.:

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main

Faloutsos CMU SCS 15-415 9



Overview - detailed

- DB design and normalization
 - pitfalls of bad design
 - decomposition
 - lossless join decomp.
 - dependency preserving
 - normal forms

Faloutsos CMU SCS 15-415 10




Decompositions

There are 'bad' decompositions. Good ones are:

- lossless and
- dependency preserving

Faloutsos CMU SCS 15-415 11



Decompositions - lossy:

R1(ssn, grade, name, address) R2(c-id, grade)

Ssn	Grade	Name	Address
123	A	smith	Main
123	B	smith	Main
234	A	jones	Forbes

c-id	Grade
413	A
415	B
211	A

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
234	211	A	jones	Forbes

ssn → name, address
ssn, c-id → grade

Faloutsos CMU SCS 15-415 12

CMU SCS

Decompositions - lossy:

can not recover original table with a join!

Ssn	Grade	Name	Address
123	A	smith	Main
123	B	smith	Main
234	A	jones	Forbes

c-id	Grade
413	A
415	B
211	A

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
234	211	A	jones	Forbes

ssn → name, address
ssn, c-id → grade

Faloutsos CMU SCS 15-415 13

CMU SCS

Decompositions

example of non-dependency preserving

S#	address	status
123	London	E
125	Paris	E
234	Pitts.	A

S#	address
123	London
125	Paris
234	Pitts.

S#	status
123	E
125	E
234	A

S# → address, status
address → status

S# → address S# → status

Faloutsos CMU SCS 15-415 14

CMU SCS

Decompositions

(drill: is it lossless?)

S#	address	status
123	London	E
125	Paris	E
234	Pitts.	A

S#	address
123	London
125	Paris
234	Pitts.

S#	status
123	E
125	E
234	A

S# → address, status
address → status

S# → address S# → status

Faloutsos CMU SCS 15-415 15

CMU SCS

Decompositions - lossless

Definition:
consider schema R, with FD 'F'. R1, R2 is a
lossless join decomposition of R if we
always have: $r1 \bowtie r2 = r$

An easier criterion?

Faloutsos CMU SCS 15-415 16

CMU SCS

Decomposition - lossless

Theorem: lossless join decomposition if the
joining attribute is a superkey in at least one
of the new tables

Formally:

$$R1 \cap R2 \rightarrow R1 \text{ or } R1 \cap R2 \rightarrow R2$$

Faloutsos CMU SCS 15-415 17

CMU SCS

Decomposition - lossless

example:

R1

Ssn	c-id	Grade
123	413	A
123	415	B
234	211	A

ssn, c-id → grade

R2

Ssn	Name	Address
123	smith	Main
234	jones	Forbes

ssn → name, address

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
234	211	A	jones	Forbes

ssn → name, address
ssn, c-id → grade

Faloutsos CMU SCS 15-415 18

CMU SCS

Overview - detailed

- DB design and normalization
 - pitfalls of bad design
 - decomposition
 - lossless join decomp.
 - dependency preserving**
 - normal forms

Faloutsos CMU SCS 15-415 19

CMU SCS

Decomposition - depend. pres.

informally: we don't want the original FDs to span two tables - counter-example:

S#	address	status
123	London	E
125	Paris	E
234	Pitts.	A

S#	address
123	London
125	Paris
234	Pitts.

S#	status
123	E
125	E
234	A

S# → address, status
address → status

S# → address S# → status

Faloutsos CMU SCS 15-415 20

CMU SCS

Decomposition - depend. pres.

dependency preserving decomposition:

S#	address	status
123	London	E
125	Paris	E
234	Pitts.	A

S#	address
123	London
125	Paris
234	Pitts.

address	status
London	E
Paris	E
Pitts.	A

S# → address, status
address → status

S# → address address → status
(but: S# → status ?)

Faloutsos CMU SCS 15-415 21

CMU SCS

Decomposition - depend. pres.

informally: we don't want the original FDs to span two tables.
More specifically: ... the FDs of the **canonical cover**.

Faloutsos CMU SCS 15-415 22

CMU SCS

Decomposition - depend. pres.

why is dependency preservation good?

S#	address	S#	status
123	London	123	E
125	Paris	125	E
234	Pitts.	234	A

S#	address	address	status
123	London	London	E
125	Paris	Paris	E
234	Pitts.	Pitts.	A

S# → address
S# → status
(address → status: 'lost')

S# → address address → status

Faloutsos CMU SCS 15-415 23

CMU SCS

Decomposition - depend. pres.

A: eg., record that 'Philly' has status 'A'


S#	address	S#	status
123	London	123	E
125	Paris	125	E
234	Pitts.	234	A

S#	address	address	status
123	London	London	E
125	Paris	Paris	E
234	Pitts.	Pitts.	A

S# → address
S# → status
(address → status: 'lost')

S# → address address → status

Faloutsos CMU SCS 15-415 24




CMU SCS

Decomposition - conclusions

- decompositions should always be lossless
 - joining attribute \rightarrow superkey
- whenever possible, we want them to be dependency preserving (occasionally, impossible - see 'STJ' example later...)

Faloutsos CMU SCS 15-415 25




CMU SCS

Overview - detailed

- DB design and normalization
 - pitfalls of bad design
 - decomposition (\rightarrow how to fix the problem)
 - **normal forms** (\rightarrow how to detect the problem)
 - BCNF,
 - 3NF
 - (1NF, 2NF)

Faloutsos CMU SCS 15-415 26



CMU SCS

Normal forms - BCNF

We saw how to fix 'bad' schemas -
but what is a 'good' schema?

Answer: 'good', if it obeys a 'normal form',
ie., a set of rules.

Typically: Boyce-Codd Normal form

Faloutsos CMU SCS 15-415 27

CMU SCS

Normal forms - BCNF

Defn.: Rel. R is in BCNF wrt F , if

- informally: everything depends on the full key, and nothing but the key
- semi-formally: every determinant (of the cover) is a candidate key

Faloutsos CMU SCS 15-415 28

CMU SCS

Normal forms - BCNF

Example and counter-example:

Ssn	Name	Address
123	smith	Main
999	smith	Shady
234	jones	Forbes

ssn \rightarrow name, address

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
234	211	A	jones	Forbes

ssn \rightarrow name, address
ssn, c-id \rightarrow grade

Faloutsos CMU SCS 15-415 29

CMU SCS

Normal forms - BCNF

Formally: for every FD $a \rightarrow b$ in F

- $a \rightarrow b$ is trivial (a superset of b) or
- a is a superkey

Faloutsos CMU SCS 15-415 30

CMU SCS

Normal forms - BCNF

Theorem: given a schema R and a set of FD 'F', we can always decompose it to schemas R_1, \dots, R_n , so that

- R_1, \dots, R_n are in BCNF and
- the decompositions are lossless.

(but, some decomp. might lose dependencies)

Faloutsos CMU SCS 15-415 31

CMU SCS

Normal forms - BCNF

How? algorithm in book: for a relation R

- for every FD $X \rightarrow A$ that violates BCNF, decompose to tables (X, A) and $(R - A)$
- repeat recursively

eg. $\text{TAKES1}(\text{ssn}, \text{c-id}, \text{grade}, \text{name}, \text{address})$

$\text{ssn} \rightarrow \text{name}, \text{address}$

$\text{ssn}, \text{c-id} \rightarrow \text{grade}$

Faloutsos CMU SCS 15-415 32

CMU SCS

Normal forms - BCNF


eg. $\text{TAKES1}(\text{ssn}, \text{c-id}, \text{grade}, \text{name}, \text{address})$

$\text{ssn} \rightarrow \text{name}, \text{address}$ $\text{ssn}, \text{c-id} \rightarrow \text{grade}$

```

graph LR
    subgraph Box1 [ ]
        ssn[ssn]
        cid[c-id]
    end
    ssn --> name[name]
    ssn --> address[address]
    Box1 --> grade[grade]
  
```

Faloutsos CMU SCS 15-415 33



CMU SCS

Normal forms - BCNF

Ssn	c-id	Grade
123	413	A
123	415	B
234	211	A

Ssn	Name	Address
123	smith	Main
123	smith	Main
234	jones	Forbes

ssn, c-id -> grade

ssn->name, address

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
234	211	A	jones	Forbes

ssn->name, address

ssn, c-id -> grade

Normal forms - BCNF

pictorially: we want a 'star' shape

```
graph LR; subgraph Box; ssn; c-id; end; ssn --> name; ssn --> address; c-id --> grade;
```

:not in BCNF

Faloutsos CMU SCS 15-415 35

CMU SCS

Normal forms - BCNF

pictorially: we want a 'star' shape

```
graph LR; A[A] --> B[B]; A --> C[C]; subgraph "or"; subgraph " "; D[D]; E[E]; end; end; " " --> F[F]; " " --> G[G]; " " --> H[H];
```

Faloutsos

CMU SCS 15-415

36

Normal forms - BCNF

or a star-like: (eg., 2 cand. keys):
STUDENT(ssn, st#, name, address)

Faloutsos CMU SCS 15-415 37

Normal forms - BCNF

but **not**:

Faloutsos CMU SCS 15-415 38

Normal forms - 3NF

consider the 'classic' case:
STJ(Student, Teacher, subJect)
T-> J
S,J -> T
is it BCNF?

Faloutsos CMU SCS 15-415 39

CMU SCS

Normal forms - 3NF

STJ(Student, Teacher, subJect)
 $T \rightarrow J$ $S, J \rightarrow T$

How to decompose it to BCNF?

Faloutsos CMU SCS 15-415 40

CMU SCS

Normal forms - 3NF

STJ(Student, Teacher, subJect)
 $T \rightarrow J$ $S, J \rightarrow T$

1) $R_1(T, J)$ $R_2(S, J)$
 (BCNF? - lossless? - dep. pres.?)

2) $R_1(T, J)$ $R_2(S, T)$
 (BCNF? - lossless? - dep. pres.?)

Faloutsos CMU SCS 15-415 41

CMU SCS

Normal forms - 3NF

STJ(Student, Teacher, subJect)
 $T \rightarrow J$ $S, J \rightarrow T$

1) $R_1(T, J)$ $R_2(S, J)$
 (BCNF? **Y+Y** - lossless? **N** - dep. pres.? **N**)

2) $R_1(T, J)$ $R_2(S, T)$
 (BCNF? **Y+Y** - lossless? **Y** - dep. pres.? **N**)

Faloutsos CMU SCS 15-415 42

CMU SCS

Normal forms - 3NF

STJ(Student, Teacher, subJect)
 $T \rightarrow J \quad S, J \rightarrow T$
 in this case: impossible to have both

- BCNF **and**
- dependency preservation

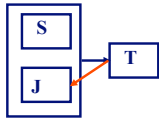
Welcome 3NF!

Faloutsos CMU SCS 15-415 43

CMU SCS

Normal forms - 3NF

STJ(Student, Teacher, subJect)
 $T \rightarrow J \quad S, J \rightarrow T$



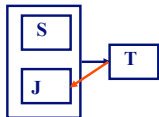
informally, 3NF
 'forgives' the arrow
 in the can. cover

Faloutsos CMU SCS 15-415 44

CMU SCS

Normal forms - 3NF


STJ(Student, Teacher, subJect)
 $T \rightarrow J \quad S, J \rightarrow T$



Formally, a rel. R with
 FDs 'F' is in 3NF if:
 for every $a \rightarrow b$ in F:

- it is trivial or
- a is a superkey or
- b : part of a candidate key

Faloutsos CMU SCS 15-415 45



CMU SCS


Normal forms - 3NF

how to bring a schema to 3NF?

two algo's in book: First one:

- start from ER diagram and turn to tables
- then we have a set of tables R_1, \dots, R_n which are in 3NF
- for each FD $(X \rightarrow A)$ in the cover that is not preserved, create a table (X, A)

Faloutsos CMU SCS 15-415 46



CMU SCS


Normal forms - 3NF

how to bring a schema to 3NF?

two algo's in book: Second one ('synthesis')

- take all attributes of R
- for each FD $(X \rightarrow A)$ in the cover, add a table (X, A)
- if not lossless, add a table with appropriate key

Faloutsos CMU SCS 15-415 47



CMU SCS

Normal forms - 3NF

Example:

$R: ABC$

$F: A \rightarrow B, C \rightarrow B$

Q1: what is the cover?

Q2: what is the decomposition to 3NF?

Faloutsos CMU SCS 15-415 48

CMU SCS

Normal forms - 3NF

Example:
R: ABC
F: $A \rightarrow B, C \rightarrow B$
Q1: what is the cover?
A1: 'F' is the cover
Q2: what is the decomposition to 3NF?

Faloutsos CMU SCS 15-415 49

CMU SCS

Normal forms - 3NF

Example:
R: ABC
F: $A \rightarrow B, C \rightarrow B$
Q1: what is the cover?
A1: 'F' is the cover
Q2: what is the decomposition to 3NF?
A2: $R_1(A,B), R_2(C,B), \dots$ [is it lossless??]

Faloutsos CMU SCS 15-415 50

CMU SCS

Normal forms - 3NF

Example:
R: ABC
F: $A \rightarrow B, C \rightarrow B$
Q1: what is the cover?
A1: 'F' is the cover
Q2: what is the decomposition to 3NF?
A2: $R_1(A,B), R_2(C,B), R_3(A,C)$

Faloutsos CMU SCS 15-415 51

CMU SCS

Normal forms - 3NF vs BCNF

- If 'R' is in BCNF, it is always in 3NF (but not the reverse)
- In practice, aim for
 - BCNF; lossless join; and dep. preservation
- if impossible, we accept
 - 3NF; but insist on lossless join and dep. preservation

Faloutsos CMU SCS 15-415 52

CMU SCS

Normal forms - more details

- why '3'NF? what is 2NF? 1NF?
- 1NF: attributes are atomic (ie., no set-valued attr., a.k.a. 'repeating groups')

Ssn	Name	Dependents
123	Smith	Peter Mary John
234	Jones	Ann Michael

not 1NF

Faloutsos CMU SCS 15-415 53

CMU SCS

Normal forms - more details

2NF: 1NF and non-key attr. fully depend on the key

counter-example: TAKES1(ssn, c-id, grade, name, address)

ssn -> name, address ssn, c-id -> grade

Faloutsos CMU SCS 15-415 54

CMU SCS

Normal forms - more details

- 3NF: 2NF and no transitive dependencies
- counter-example:

in 2NF, but **not** in 3NF

Faloutsos CMU SCS 15-415 55

CMU SCS

Normal forms - more details

- 4NF, multivalued dependencies etc:
IGNORE
- in practice, E-R diagrams usually lead to
tables in BCNF

Faloutsos CMU SCS 15-415 56

CMU SCS

Overview - conclusions

DB design and normalization

- pitfalls of bad design
- decompositions (lossless, dep. preserving)
- normal forms (BCNF or 3NF)

“everything should depend on the key, the **whole** key, and **nothing but** the key”

Faloutsos CMU SCS 15-415 57
