Comp 115: Databases

Decomposition & Schema Normalization

Instructor: Manos Athanassoulis

http://www.cs.tufts.edu/comp/115/

Review: Database Design

Requirements Analysis

user needs; what must database do?

Conceptual Design

high level description (often done w/ ER model)

Logical Design

translate ER into DBMS data model

Schema Refinement

consistency, normalization

Physical Design

indexes, disk layout

Why schema refinement

what is a bad schema?

a schema with redundancy!

why?

redundant storage & insert/update/delete anomalies

how to fix it? 

normalize the schema by decomposing

normal forms: BCNF, 3NF, …

Motivating Example

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	John	25K	617-555-3761

SSN $\rightarrow$ Name, Salary

Motivating Example

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	John	25K	617-555-3761

SSN $\rightarrow$ Name, Salary

SSN	Name	Salary
987-00-8761	John	65K
123-00-9876	Anna	80K
787-00-4321	John	25K

SSN	Telephone
987-00-8761	857-555-1234
987-00-8761	857-555-8800
123-00-9876	617-555-9876
787-00-4321	617-555-3761

Motivating Example 2

name	category	color	price	department
iPhone	smartphone	black	600	phones
Lenovo Yoga	laptop	grey	800	computers
unifi	networking	white	150	computers
unifi	cables	white	10	stationary
OnePlus	smartphone	silver	450	phones

name, category $\rightarrow$ price, color

category $\rightarrow$ department

Motivating Example 2

name	category	color	price	department
iPhone	smartphone	black	600	phones
Lenovo Yoga	laptop	grey	800	computers
unifi	networking	white	150	computers
unifi	cables	white	10	stationary
OnePlus	smartphone	silver	450	phones

name	category	color	price
iPhone	smartphone	black	600
Lenovo Yoga	laptop	grey	800
unifi	networking	white	150
unifi	cables	white	10
OnePlus	smartphone	silver	450

category	department
laptop	computers
networking	computers
cables	stationary
smartphone	phones

“chopping the relation into pieces using FDs”

DECOMPOSITION

Decomposition

Formally

we decompose **R**$(A_1, \ldots, A_n)$ by creating:

R$_1$ $(B_1, \ldots, B_m)$

R$_2$ $(C_1, \ldots, C_k)$

where $\{B_1, \ldots, B_m\} \cup \{C_1, \ldots, C_k\} = \{A_1, \ldots, A_n\}$

the instance of ***R$_1$*** is the <u>projection</u> of **R** onto $B_1, \ldots, B_m$

the instance of ***R$_2$*** is the <u>projection</u> of **R** onto $C_1, \ldots, C_k$

Motivating Example

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	John	25K	617-555-3761

SSN	Name	Salary
987-00-8761	John	65K
123-00-9876	Anna	80K
787-00-4321	John	25K

SSN	Telephone
987-00-8761	857-555-1234
987-00-8761	857-555-8800
123-00-9876	617-555-9876
787-00-4321	617-555-3761

“Good” Decomposition

(1) minimize redundancy

(2) avoid information loss (***lossless-join***)

(3) preserve FDs (***dependency preserving***)

(4) ensure good query performance

Information Loss

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	John	25K	617-555-3761

Decompose into:
R_1(SSN, Name, Salary)
R_2(Name, Telephone)

SSN	Name	Salary
987-00-8761	John	65K
123-00-9876	Anna	80K
787-00-4321	John	25K

Name	Telephone
John	857-555-1234
John	857-555-8800
Anna	617-555-9876
John	617-555-3761

can we reconstruct R?

Lossless Decomposition

$$\begin{tikzcd} & \mathbf{R}(A,B,C) \arrow[dl] \arrow[dr, "\textit{decompose}"] & \\ \mathbf{R_1}(A,B) \arrow[dr] & & \mathbf{R_2}(B,C) \arrow[dl, "\textit{recover (join on B)}"] \\ & \mathbf{R'}(A,B,C) & \end{tikzcd}$$

the decomposition is *lossless-join* if
for any initial instance ***R*, *R* = *R'***

Lossless Criterion

given a relation **R**(A) and a set *F* of FDs

and a decomposition of **R** into $\mathbf{R_1}(A_1)$ and $\mathbf{R_2}(A_2)$

the decomposition is *lossless-join* **if and only if**

at least one of the FDs is in F^+ (closure of F):

(1) $A_1 \cap A_2 \rightarrow A_1$

(2) $A_1 \cap A_2 \rightarrow A_2$

Example

Relation **R**(A, B, C, D)

FD $A \rightarrow B, C$

lossy

decomposition into $\mathbf{R_1}$(A, B, C) and $\mathbf{R_2}$(D)

$A_1 \cap A_2$ empty set

lossless-join 

decomposition into $\mathbf{R_1}$(A, B, C) and $\mathbf{R_2}$(A, D)

$A_1 \cap A_2 = A$ and $A_1 = A, B, C$

$A \rightarrow A, B, C$ *is in* F^+

Dependency Preserving

given **R** and a set of FDs *F*, we decompose **R** into $\mathbf{R_1}$ and $\mathbf{R_2}$. Suppose:

$\mathbf{R_1}$ has a set of FDs F_1
$\mathbf{R_2}$ has a set of FDs F_2
F_1 and F_2 are computed from F

it is <u>dependency preserving</u> if by enforcing F_1 over $\mathbf{R_1}$ and F_2 over $\mathbf{R_2}$, we can enforce *F* over **R**

(Good) Example

Person (SSN, name, age, canDrink)

SSN $\rightarrow$ *name, age*

age $\rightarrow$ *canDrink*

what is a ***dependency preserving*** *decomposition?*

R_1(SSN, name, age) and **R_2**(age, canDrink)

SSN $\rightarrow$ *name, age* *age* $\rightarrow$ *canDrink*

Is it also lossless-join?

Yes! $A_1 \cap A_2$ = age and A_2 = age, canDrink

age $\rightarrow$ age, canDrink *is in* F^+

(Bad) Example

R (A, B, C)

$A \rightarrow B$

$B, C \rightarrow A$

A	B
a_1	b
a_2	b

A	C
a_1	c
a_2	c

A	B	C
a_1	b	c
a_2	b	c

not dependency preserving

the table violates
$B, C \rightarrow A$

$\mathbf{R_1}$(A, B) and $\mathbf{R_2}$(A, C)

$A \rightarrow B$ *no FDs!*

Normal Forms

How "good" is a schema design?
follows normal forms

flat tables
atomic values

1NF
2NF
3NF
BCNF
4NF
…

more
restrictive

Normal Forms

How “good” is a schema design?
follows normal forms

flat tables
atomic values

1NF
2NF
3NF
BCNF
4NF
…

more
restrictive

Boyce-Codd Normal Form (BCNF)

given a relation **R**($A_1,\ldots,A_n$),
a set of FDs *F*, and $X \subseteq \{A_1,\ldots,A_n\}$
R is in BCNF if $\forall$ $X \rightarrow A$ one of the two holds:

- $A \in X$ (that is, it is a trivial FD)
- X is a <u>superkey</u>

[alternatively, $\forall$ non-trivial FD $X \rightarrow A$,
X is a superkey in R]

BCNF - Example

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	John	25K	617-555-3761

SSN → Name, Salary

key: {SSN, Telephone}

FD is not trivial!

so, is SSN a superkey?

*no! it is **not** in **BCNF***

BCNF - Example 2

SSN	Name	Salary
987-00-8761	John	65K
123-00-9876	Anna	80K
787-00-4321	John	25K

SSN → Name, Salary

key: {SSN}

FD is not trivial!

so, is SSN a superkey?

*yes! it is in **BCNF***

BCNF - Example 3

SSN	Telephone
987-00-8761	857-555-1234
987-00-8761	857-555-8800
123-00-9876	617-555-9876
787-00-4321	617-555-3761

key: {SSN, Telephone} *the relation is in* ***BCNF***

why?

*Is it possible a binary relation to <u>not</u> be in **BCNF**?*

no FDs

Binary Relations always BCNF

R (A,B)

excluding all trivial FDs, there are three cases:

(1) **R** has no FD

(2) **R** has one FD, either *A* $\rightarrow$ *B or B* $\rightarrow$ *A, or,*

(3) **R** has two FDs, *A* $\rightarrow$ *B and B* $\rightarrow$ *A*

(1) trivially in BCNF

(2) in either LHS is the key (hence, superkey)

(3) both, A and B candidate keys

BCNF Decomposition

Find a FD that violates BCNF:

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

Decompose **R** to **R_1** and **R_2**

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

R_2 $(A_1, \ldots, A_n,$ all other attributes$)$

continue until no BCNF violations are left

(in new tables as well)

Our favorite example!

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	John	25K	617-555-3761

SSN $\rightarrow$ *Name, Salary* violates BCNF

A_1 = SSN, B_1 = *Name*, B_2 = *Salary*

Split in two relations:

R_1(SSN, Name, Salary)

R_2(SSN, Telephone)

Our favorite example!

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	John	25K	617-555-3761

SSN	Name	Salary
987-00-8761	John	65K
123-00-9876	Anna	80K
787-00-4321	John	25K

SSN	Telephone
987-00-8761	857-555-1234
987-00-8761	857-555-8800
123-00-9876	617-555-9876
787-00-4321	617-555-3761

BCNF Decomposition Properties

removes [certain types of] redundancy

is lossless-join

is <u>not always</u> dependency preserving

BCNF – Lossless Join

Example

R (A, B, C) and FD: $A \rightarrow B$

superkey(s) of the relation?

$\{A, C\}^+, \{A, B, C\}^+ = \{A, B, C\}$

$A \rightarrow B$ *violates* BCNF (A is not a superkey)

so, the BCNF decomposition is :

$\mathbf{R_1}$(A, B) and $\mathbf{R_2}$(A, C)

we can reconstruct it!

BCNF – **not** dependency preserving

Example

R (A, B, C), FDs: $A \rightarrow B$ and $B, C \rightarrow A$

superkey(s) of the relation?

$\{A, C\}^+, \{B, C\}^+, \{A, B, C\}^+ = \{A, B, C\}$

$B, C \rightarrow A$ is ok, but $A \rightarrow B$ violates BCNF

so, the BCNF decomposition is :

$\mathbf{R_1}$ (A, B) and $\mathbf{R_2}$ (A, C)

$A \rightarrow B$ is preserved in $\mathbf{R_1}$

$B, C \rightarrow A$ is not preserved!

BCNF Decomposition Examples

Books (author, gender, booktitle, genre, price)

author $\rightarrow$ *gender*

booktitle $\rightarrow$ *genre, price*

candidate key(s)? 

{author, booktitle} is the only one

Is it in BCNF? 

No, because LHS of both FD are not a superkey!

BCNF Decomposition Examples

Books (author, gender, booktitle, genre, price)
author → gender
booktitle → genre, price

Splitting using: *author → gender*

AuthorInfo (author, gender)
FD author → gender (in BCNF!)

Book2 (author, booktitle, genre, price)
FD booktitle → genre, price
is booktitle a superkey?

No! {booktitle, author} is.
So not in BCNF!

BCNF Decomposition Examples

Books (author, gender, booktitle, genre, price)
author → *gender*
booktitle → *genre, price*

AuthorInfo (author, gender)

Further splitting with *booktitle* → *genre, price*
Book2 (author, booktitle, genre, price)

BookAuthor (booktitle, author) binary is in BCNF!
BookInfo (booktitle, genre, price) in BCNF!
FD booktitle → *genre, price*
is booktitle a superkey?

Yes!

what if not dependency preserving?

in some cases BCNF decomposition is not dependency preserving

how to address this? 

relax the normalization requirements

Third Normal Form (3NF)

given a relation **R** $(A_1,\ldots,A_n)$,
a set of FDs *F*, and $X \subseteq \{A_1,\ldots,A_n\}$
R is in 3NF if $\forall\ X \rightarrow A$ one of the three holds:

- $A \in X$ (that is, it is a trivial FD)
- X is a <u>superkey</u>
- A is <u>part of some key</u> for **R**

is a relation in 3NF also in BCNF? 

No, but a relation in BCNF is always in 3NF!

Third Normal Form (3NF)

Example

R (A, B, C), FDs $C \rightarrow A$ and $A, B \rightarrow C$
is in 3NF but not in BCNF. Why?

superkeys?
{A, B}, {B, C}, and {A, B, C}

candidate keys?
{A, B} and {B, C}

Compromise: aim for BCNF but settle for 3NF
lossless-join & dependency preserving possible

3NF Algorithm

(1) apply BCNF until all relations are in 3NF

(2) compute a minimal cover *F'* of *F*

(3) for each non-preserved FD X $\rightarrow$ A in *F'*, add a new relation **R** (X, A)

3NF algorithm example

Assume **R** (A, B, C, D)

$A \rightarrow D$

$A, B \rightarrow C$

$A, D \rightarrow C$

$B \rightarrow C$

$D \rightarrow A, B$

superkeys?

{A} {D} {A, B} {A, D}, …

not {B}

Step 1: find a BCNF decomposition

R_1 (B, C)

R_2 (A, B, D)

3NF algorithm example

Assume **R** (A, B, C, D)

$A \rightarrow D$

$A, B \rightarrow C$

$A, D \rightarrow C$

$B \rightarrow C$

$D \rightarrow A, B$

Step 2: find a minimal cover

$A \rightarrow D$

$B \rightarrow C$

$D \rightarrow A$

$D \rightarrow B$

3NF algorithm example

Assume **R** (A, B, C, D)

$A \rightarrow D$

$A, B \rightarrow C$

$A, D \rightarrow C$

$B \rightarrow C$

$D \rightarrow A, B$

Step 3: add a new relation for not preserved FDs

$A \rightarrow D$

$B \rightarrow C$

$D \rightarrow A$

$D \rightarrow B$

R_1 (B, C)

R_2 (A, B, D)

all FD are preserved!

both are in BCNF!

Is Normalization Always Good?

Example 1: suppose A and B are always used together, but normalization says they should be in different tables (e.g., hours_worked and hourly_rate)

decomposition might produce unacceptable performance loss

Example 2: data warehouses

huge historical DBs, rarely updated after creation

joins expensive or impractical

[we want “flat” tables, a.k.a, denormalized]

Example

R (C, S, J, D, P, Q, V)

C $\rightarrow$ S, J, D, P, Q, V

J, P $\rightarrow$ C

S, D $\rightarrow$ P

J $\rightarrow$ S

Step 1:

R_1 (S, D, P)

R_2 (C, S, J, D, Q, V)

superkeys?

{C}, {J, P}, {D, J}, …

not {S, D}

Example

R (C, S, J, D, P, Q, V)   superkeys? 

C → S, J, D, P, Q, V   {C}, {J, P}, {D, J}, …

J, P → C   not {S, D}

S, D → P   superkeys of $\mathbf{R_2}$ (C, S, J, D, Q, V) ? 

J → S   {C}, … not {J}

Step 1b:

$\mathbf{R_1}$ (S, D, P)

$\mathbf{R_{2'}}$ (J, S)

$\mathbf{R_3}$ (C, J, D, Q, V)

Example

R (C, S, J, D, P, Q, V)

C $\rightarrow$ S, J, D, P, Q, V

J, P $\rightarrow$ C

S, D $\rightarrow$ P

J $\rightarrow$ S

R_1 (S, D, P)

$R_{2'}$ (J, S)

R_3 (C, J, D, Q, V)

R_4 (J, P, C)

Step 2: Minimal Cover

C $\rightarrow$ J, C $\rightarrow$ D, C $\rightarrow$ Q, C $\rightarrow$ V

J, P $\rightarrow$ C

S, D $\rightarrow$ P

J $\rightarrow$ S

are they all preserved?

No!

Step 3: need to add **R_4** (J, P, C)

Example

R (C, S, J, D, P, Q, V)

$C \rightarrow S, J, D, P, Q, V$

$J, P \rightarrow C$

$S, D \rightarrow P$

$J \rightarrow S$

R_1 (S, D, P)

$R_{2'}$ (J, S)

R_3 (C, J, D, Q, V)

R_4 (J, P, C)

Step 2: Minimal Cover

$C \rightarrow J$, $C \rightarrow D$, $C \rightarrow Q$, $C \rightarrow V$

$J, P \rightarrow C$

$S, D \rightarrow P$

$J \rightarrow S$

are they all preserved?

No!

Step 3: need to add **R_4** (J, P, C)

did we just introduce redundancy?

Lesson!

theory of normalization is a guide

<u>cannot always</u> give a "perfect" solution

redundancy

alternatives

query performance

Summary

fix bad schemas (redundancy) by decomposition

lossless-join

dependency preserving

Desired normal forms

BCNF: only superkey FDs

3NF: superkey FDs + dependencies to prime attributes in RHS

Next: execution of queries

procedural & declarative