Introduction to Data Management CSE 344

Lectures 18-19: Design Theory

Announcements

- HW5 is due tonight
- HW6 is out and due next Friday
  - Mostly pen-and-paper with some SQL
- Webquiz 6 due next Wednesday
- Today and next lecture:
  - Design theory (3.1-3.4)

Office hour canceled this Wednesday

Database Design Process

Conceptual Model:

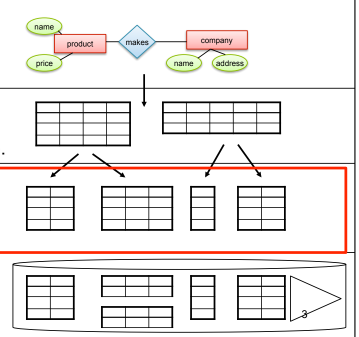
Relational Model:
Tables + constraints
And also functional dep.

Normalization:
Eliminates anomalies

Conceptual Schema

Physical storage details

Physical Schema



Relational Schema Design

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield

One person may have multiple phones, but lives in only one city

Primary key is thus (SSN, PhoneNumber)

What is the problem with this schema?

Relational Schema Design

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield

Anomalies:
- Redundancy = repeat data
- Update anomalies = what if Fred moves to “Bellevue”?
- Deletion anomalies = what if Joe deletes his phone number?

Relation Decomposition

Break the relation into two:

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield

Name	SSN	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Westfield

SSN	PhoneNumber
123-45-6789	206-555-1234
123-45-6789	206-555-6543
987-65-4321	908-555-2121

Anomalies have gone:
- No more repeated data
- Easy to move Fred to “Bellevue” (how ?)
- Easy to delete all Joe’s phone numbers (how ?)

Relational Schema Design (or Logical Design)

How do we do this systematically?

- Start with some relational schema
- Find out its ***functional dependencies*** (FDs)
- Use FDs to ***normalize*** the relational schema

Functional Dependencies (FDs)

Definition

If two tuples agree on the attributes

$A_1, A_2, \ldots, A_n$

then they must also agree on the attributes

$B_1, B_2, \ldots, B_m$

Formally:

$A_1 \ldots A_n$ **determines** $B_1 .. B_m$

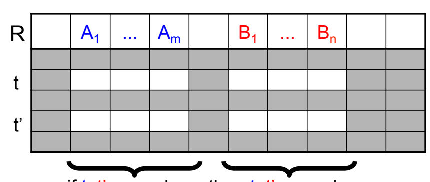
$A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m$

Functional Dependencies (FDs)

Definition $A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n$ **holds** in R if:

$\forall t, t' \in R,$

$(t.A_1 = t'.A_1 \wedge \ldots \wedge t.A_m = t'.A_m \rightarrow t.B_1 = t'.B_1 \wedge \ldots \wedge t.B_n = t'.B_n)$



if t, t' agree here then t, t' agree here

Example

An FD holds, or does not hold on an instance:

EmpID	**Name**	**Phone**	**Position**
E0045	Smith	1234	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

EmpID $\rightarrow$ Name, Phone, Position

Position $\rightarrow$ Phone

but not Phone $\rightarrow$ Position

Example

EmpID	**Name**	**Phone**	**Position**
E0045	Smith	1234	Clerk
E3542	Mike	9876 ←	Salesrep
E1111	Smith	9876 ←	Salesrep
E9999	Mary	1234	Lawyer

Position $\rightarrow$ Phone

Example

EmpID	**Name**	**Phone**	**Position**
E0045	Smith	1234 →	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234 →	Lawyer

But not Phone $\rightarrow$ Position

Example

name → color
category → department
color, category → price

name	category	color	department	price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Green	Toys	99

Do all the FDs hold on this instance?

Example

name → color
category → department
color, category → price

name	category	color	department	price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Green	Toys	49
Gizmo	Stationary	Green	Office-supp.	59

What about this one ?

Terminology

- FD **holds** or **does not hold** on an instance
- If we can be sure that *every instance of R* will be one in which a given FD is true, then we say that **R satisfies the FD**
- If we say that R satisfies an FD F, we are **stating a constraint on R**

An Interesting Observation

If all these FDs are true:

name → color
category → department
color, category → price

Then this FD also holds: name, category → price

If we find out from application domain that a relation satisfies some FDs, it doesn't mean that we found all the FDs that it satisfies!
There could be more FDs implied by the ones we have.

Closure of a set of Attributes

Given a set of attributes $A_1, \ldots, A_n$

The **closure**, $\{A_1, \ldots, A_n\}^+$ = the set of attributes B s.t. $A_1, \ldots, A_n \rightarrow B$

Example:

1. name → color
2. category → department
3. color, category → price

Closures:

name$^+$ = {name, color}
{name, category}$^+$ = {name, category, color, department, price}
color$^+$ = {color}

Closure Algorithm

X={A1, …, An}.

Repeat until X doesn't change **do**:
if $B_1, \ldots, B_n \rightarrow C$ is a FD **and** $B_1, \ldots, B_n$ are all in X
then add C to X.

Example:

1. name → color
2. category → department
3. color, category → price

{name, category}$^+$ =
{ name, category, color, department, price }

Hence: name, category → color, department, price

Example

In class:

R(A,B,C,D,E,F)

A, B → C
A, D → E
B → D
A, F → B

Compute $\{A,B\}^+$ X = {A, B, }

Compute $\{A, F\}^+$ X = {A, F, }

Example

In class:

R(A,B,C,D,E,F)

A, B → C
A, D → E
B → D
A, F → B

Compute $\{A,B\}^+$ X = {A, B, C, D, E }

Compute $\{A, F\}^+$ X = {A, F, }

Example

In class:

R(A,B,C,D,E,F)

A, B → C
A, D → E
B → D
A, F → B

Compute $\{A,B\}^+$ X = {A, B, C, D, E }

Compute $\{A, F\}^+$ X = {A, F, B, C, D, E }

Example

In class:

R(A,B,C,D,E,F)

A, B → C
A, D → E
B → D
A, F → B

Compute $\{A,B\}^+$ X = {A, B, C, D, E }

Compute $\{A, F\}^+$ X = {A, F, B, C, D, E }

What is the key of R?

Practice at Home

Find all FD's implied by:

A, B → C
A, D → B
B → D

Practice at Home

Find all FD's implied by:

A, B → C
A, D → B
B → D

Step 1: Compute X^+, for every X:

A+ = A, B+ = BD, C+ = C, D+ = D
AB+ =ABCD, AC+=AC, AD+=ABCD,
BC+=BCD, BD+=BD, CD+=CD
ABC+ = ABD+ = ACD$^+$ = ABCD (no need to compute– why ?)
BCD$^+$ = BCD, ABCD+ = ABCD

Step 2: Enumerate all FD's X → Y, s.t. $Y \subseteq X^+$ and $X \cap Y = \varnothing$:

AB → CD, AD→BC, ABC → D, ABD → C, ACD → B

Keys

- A **superkey** is a set of attributes $A_1, ..., A_n$ s.t. for any other attribute B, we have $A_1, ..., A_n \rightarrow B$
- A **key** is a minimal superkey
  - A superkey and for which no subset is a superkey

Computing (Super)Keys

- For all sets X, compute X^+
- If X^+ = [all attributes], then X is a superkey
- Try only the minimal X's to get the key

Example

Product(name, price, category, color)

name, category → price
category → color

What is the key ?

Example

Product(name, price, category, color)

name, category → price
category → color

What is the key ?

(name, category) + = { name, category, price, color }

Hence (name, category) is a key

Key or Keys ?

Can we have more than one key ?

Given R(A,B,C) define FD's s.t. there are two or more keys

Key or Keys ?

Can we have more than one key ?

Given R(A,B,C) define FD's s.t. there are two or more keys

A → B
B → C
C → A

or

AB→C
BC→A

or

A→BC
B→AC

what are the keys here ?

Eliminating Anomalies

Main idea:

- X → A is OK if X is a (super)key
- X → A is not OK otherwise
  - Need to decompose the table, but how?

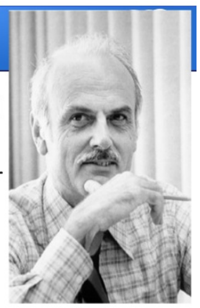
Boyce-Codd Normal Form

Boyce-Codd Normal Form

Dr. Raymond F. Boyce

Edgar Frank "Ted" Codd

"A Relational Model of Data for Large Shared Data Banks"



Boyce-Codd Normal Form

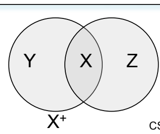
There are no "bad" FDs:

Definition. A relation R is in BCNF if:
Whenever X→ B is a non-trivial dependency, then X is a superkey.

Equivalently: **Definition**. A relation R is in BCNF if:
$\forall$ X, either $X^+ = X$ or X^+ = [all attributes]

BCNF Decomposition Algorithm

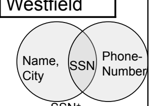
Normalize(R)
  find X s.t.: $X \neq X^+$ and $X^+ \neq$ [all attributes]
  if (not found) **then** "R is in BCNF"
  let $Y = X^+ - X$; Z = [all attributes] - X^+
  decompose R into R1($X \cup Y$) and R2($X \cup Z$)
  Normalize(R1); Normalize(R2);



Example

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield
Joe	987-65-4321	908-555-1234	Westfield

SSN → Name, City



The only key is: {SSN, PhoneNumber}
Hence SSN → Name, City is a "bad" dependency

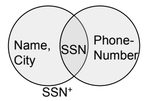
In other words:
SSN+ = SSN, Name, City and is neither SSN nor All Attributes

Example BCNF Decomposition

Name	SSN	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Westfield

SSN → Name, City

SSN	PhoneNumber
123-45-6789	206-555-1234
123-45-6789	206-555-6543
987-65-4321	908-555-2121
987-65-4321	908-555-1234



Let's check anomalies:
- Redundancy ?
- Update ?
- Delete ?

Find X s.t.: X ≠X⁺ and X⁺ ≠ [all attributes]

Example BCNF Decomposition

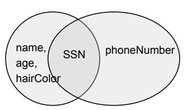
Person(name, SSN, age, hairColor, phoneNumber)
SSN → name, age
age → hairColor

Find X s.t.: X ≠X⁺ and X⁺ ≠ [all attributes]

Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)
SSN → name, age
age → hairColor

Iteration 1: Person: SSN+ = SSN, name, age, hairColor
Decompose into: P(SSN, name, age, hairColor)
Phone(SSN, phoneNumber)



Find X s.t.: X ≠X⁺ and X⁺ ≠ [all attributes]

Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)
SSN → name, age
age → hairColor

What are the keys ?

Iteration 1: Person: SSN+ = SSN, name, age, hairColor
Decompose into: P(SSN, name, age, hairColor)
Phone(SSN, phoneNumber)

Iteration 2: P: age+ = age, hairColor
Decompose: People(SSN, name, age)
Hair(age, hairColor)
Phone(SSN, phoneNumber)

Find X s.t.: X ≠X⁺ and X⁺ ≠ [all attributes]

Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)
SSN → name, age
age → hairColor

Note the keys!

Iteration 1: Person: SSN+ = SSN, name, age, hairColor
Decompose into: P(<u>SSN</u>, name, age, hairColor)
Phone(<u>SSN, phoneNumber</u>)

Iteration 2: P: age+ = age, hairColor
Decompose: People(<u>SSN</u>, name, age)
Hair(<u>age</u>, hairColor)
Phone(<u>SSN, phoneNumber</u>)

R(A,B,C,D)

Example: BCNF

A → B
B → C

R(A,B,C,D)

R(A,B,C,D)

Example: BCNF

$A \rightarrow B$
$B \rightarrow C$

Recall: find X s.t. $X \subsetneq X^+ \subsetneq$ [all-attrs]

R(A,B,C,D)

R(A,B,C,D)

Example: BCNF

$A \rightarrow B$
$B \rightarrow C$

R(A,B,C,D)
$A^+ = ABC \neq ABCD$

R(A,B,C,D)

Example: BCNF

$A \rightarrow B$
$B \rightarrow C$

R(A,B,C,D)
$A^+ = ABC \neq ABCD$

$R_1(A,B,C)$ $R_2(A,D)$

R(A,B,C,D)

Example: BCNF

$A \rightarrow B$
$B \rightarrow C$

R(A,B,C,D)
$A^+ = ABC \neq ABCD$

$R_1(A,B,C)$
$B^+ = BC \neq ABC$

$R_2(A,D)$

R(A,B,C,D)

Example: BCNF

$A \rightarrow B$
$B \rightarrow C$

R(A,B,C,D)
$A^+ = ABC \neq ABCD$

$R_1(A,B,C)$
$B^+ = BC \neq ABC$

$R_2(A,D)$

$R_{11}(B,C)$ $R_{12}(A,B)$

What are the keys ?

What happens if in R we first pick B^+ ? Or AB^+ ?

Decompositions in General

$R(A_1, ..., A_n, B_1, ..., B_m, C_1, ..., C_p)$

$S_1(A_1, ..., A_n, B_1, ..., B_m)$ $S_2(A_1, ..., A_n, C_1, ..., C_p)$

S_1 = projection of R on $A_1, ..., A_n, B_1, ..., B_m$
S_2 = projection of R on $A_1, ..., A_n, C_1, ..., C_p$

Lossless Decomposition

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Name	Price
Gizmo	19.99
OneClick	24.99
~~Gizmo~~	~~19.99~~

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Lossy Decomposition

What is lossy here?

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

Lossy Decomposition

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

Decomposition in General

$R(A_1, ..., A_n, B_1, ..., B_m, C_1, ..., C_p)$

$S_1(A_1, ..., A_n, B_1, ..., B_m)$ $S_2(A_1, ..., A_n, C_1, ..., C_p)$

Let: S_1 = projection of R on $A_1, ..., A_n, B_1, ..., B_m$
S_2 = projection of R on $A_1, ..., A_n, C_1, ..., C_p$

The decomposition is called *lossless* if $R = S_1 \bowtie S_2$

Fact: If $A_1, ..., A_n \rightarrow B_1, ..., B_m$ then the decomposition is lossless

It follows that every BCNF decomposition is lossless

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

$R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)$
R satisfies: $A \rightarrow B$, $B \rightarrow C$, $CD \rightarrow A$

$S1 = \Pi_{AD}(R)$, $S2 = \Pi_{AC}(R)$, $S3 = \Pi_{BCD}(R)$,
hence $R \subseteq S1 \bowtie S2 \bowtie S3$
Need to check: $R \supseteq S1 \bowtie S2 \bowtie S3$

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

$R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)$
R satisfies: $A \rightarrow B$, $B \rightarrow C$, $CD \rightarrow A$

$S1 = \Pi_{AD}(R)$, $S2 = \Pi_{AC}(R)$, $S3 = \Pi_{BCD}(R)$,
hence $R \subseteq S1 \bowtie S2 \bowtie S3$
Need to check: $R \supseteq S1 \bowtie S2 \bowtie S3$
Suppose $(a,b,c,d) \in S1 \bowtie S2 \bowtie S3$ Is it also in R?
R must contain the following tuples:

A	B	C	D	
a	b1	c1	d	Why ? $(a,d) \in S1 = \Pi_{AD}(R)$

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A→B, B→C, CD→A

S1 = Π_{AD}(R), S2 = Π_{AC}(R), S3 = Π_{BCD}(R),
hence R ⊆ S1 ⋈ S2 ⋈ S3
Need to check: R ⊇ S1 ⋈ S2 ⋈ S3
Suppose (a,b,c,d) ∈ S1 ⋈ S2 ⋈ S3 Is it also in R?
R must contain the following tuples:

A	B	C	D	Why ?
a	b1	c1	d	(a,d) ∈S1 = Π_{AD}(R)
a	b2	c	d2	(a,c) ∈S2 = Π_{BD}(R)

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A→B, B→C, CD→A

S1 = Π_{AD}(R), S2 = Π_{AC}(R), S3 = Π_{BCD}(R),
hence R ⊆ S1 ⋈ S2 ⋈ S3
Need to check: R ⊇ S1 ⋈ S2 ⋈ S3
Suppose (a,b,c,d) ∈ S1 ⋈ S2 ⋈ S3 Is it also in R?
R must contain the following tuples:

A	B	C	D	Why ?
a	b1	c1	d	(a,d) ∈S1 = Π_{AD}(R)
a	b2	c	d2	(a,c) ∈S2 = Π_{BD}(R)
a3	b	c	d	(b,c,d) ∈S3 = Π_{BCD}(R)

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A→B, B→C, CD→A

S1 = Π_{AD}(R), S2 = Π_{AC}(R), S3 = Π_{BCD}(R),
hence R ⊆ S1 ⋈ S2 ⋈ S3
Need to check: R ⊇ S1 ⋈ S2 ⋈ S3
Suppose (a,b,c,d) ∈ S1 ⋈ S2 ⋈ S3 Is it also in R?
R must contain the following tuples:

A	B	C	D	Why ?
a	b1	c1	d	(a,d) ∈S1 = Π_{AD}(R)
a	b2	c	d2	(a,c) ∈S2 = Π_{BD}(R)
a3	b	c	d	(b,c,d) ∈S3 = Π_{BCD}(R)

"Chase" them (apply FDs):

A→B

A	B	C	D
a	b1	c1	d
a	b1	c	d2
a3	b	c	d

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A→B, B→C, CD→A

S1 = Π_{AD}(R), S2 = Π_{AC}(R), S3 = Π_{BCD}(R),
hence R ⊆ S1 ⋈ S2 ⋈ S3
Need to check: R ⊇ S1 ⋈ S2 ⋈ S3
Suppose (a,b,c,d) ∈ S1 ⋈ S2 ⋈ S3 Is it also in R?
R must contain the following tuples:

A	B	C	D	Why ?
a	b1	c1	d	(a,d) ∈S1 = Π_{AD}(R)
a	b2	c	d2	(a,c) ∈S2 = Π_{BD}(R)
a3	b	c	d	(b,c,d) ∈S3 = Π_{BCD}(R)

"Chase" them (apply FDs):

A→B

A	B	C	D
a	b1	c1	d
a	b1	c	d2
a3	b	c	d

B→C

A	B	C	D
a	b1	c	d
a	b1	c	d2
a3	b	c	d

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A→B, B→C, CD→A

S1 = Π_{AD}(R), S2 = Π_{AC}(R), S3 = Π_{BCD}(R),
hence R ⊆ S1 ⋈ S2 ⋈ S3
Need to check: R ⊇ S1 ⋈ S2 ⋈ S3
Suppose (a,b,c,d) ∈ S1 ⋈ S2 ⋈ S3 Is it also in R?
R must contain the following tuples:

A	B	C	D	Why ?
a	b1	c1	d	(a,d) ∈S1 = Π_{AD}(R)
a	b2	c	d2	(a,c) ∈S2 = Π_{BD}(R)
a3	b	c	d	(b,c,d) ∈S3 = Π_{BCD}(R)

"Chase" them (apply FDs):

A→B

A	B	C	D
a	b1	c1	d
a	b1	c	d2
a3	b	c	d

B→C

A	B	C	D
a	b1	c	d
a	b1	c	d2
a3	b	c	d

CD→A

A	B	C	D
a	b1	c	d
a	b1	c	d2
a	b	c	d

Hence R contains (a,b,c,d)

Schema Refinements = Normal Forms

- 1st Normal Form = all tables are flat
- 2nd Normal Form = obsolete
- Boyce Codd Normal Form = no bad FDs
- 3rd Normal Form = see book
  - BCNF is lossless but can cause loss of ability to check some FDs (see book 3.4.4)
  - 3NF fixes that (is lossless and dependency-preserving), but some tables might not be in BCNF – i.e., they may have redundancy anomalies

CSE 344 - Winter 2016 60