Administrivia

Next Monday is a holiday

GO VOTE!

L17
Normalization is a Good Idea
Continued

Let's order pizza

One type of meat, cheese, and vegetable

Pizza	Topping	Type
1	Mozzarella	Cheese
1	Pepperoni	Meat
1	Olives	Vegetable
2	Mozzarella	Cheese
2	Sausage	Meat
2	Peppers	Vegetable

Key? (Pizza, Type)

Pizza: Dependencies?

Pizza	Topping	Type
1	Mozzarella	Cheese
1	Pepperoni	Meat
1	Olives	Vegetable
2	Mozzarella	Cheese
2	Sausage	Meat
2	Peppers	Vegetable

Topping → Type
Pizza, Type → Topping

Is this in BCNF?

Pizza: Decomposition?

Pizza	Topping
1	Mozzarella
1	Pepperoni
1	Olives
2	Mozzarella
2	Sausage
2	Peppers

Topping	Type
Mozzarella	Cheese
Pepperoni	Meat
Olives	Vegetable
Sausage	Meat
Peppers	Vegetable

Topping → Type
Pizza, Type → Topping : Lost this dependency!
(In SQL: Can't enforce one topping type)

BCNF in general

Decomposition may not preserve dependencies

In practice: additional checks may be needed
e.g. join to enforce topping type constraint

3rd Normal Form (3NF)

Relax BCNF (e.g., BCNF⊆3NF)

```
F: set of functional dependencies over relation R
    for (X→Y) in F
        Y is in X OR
        X is a superkey of R
```

3rd Normal Form (3NF)

Relax BCNF (e.g., BCNF⊆3NF)

```
F: set of functional dependencies over relation R
    for (X→Y) in F
        Y is in X OR
        X is a superkey of R OR
        Y is part of a key in R
```

Is new condition trivial? NO! key is minimal

Nice properties

lossless join ^ dependency preserving decomposition to 3NF always possible

Pizza: Dependencies?

Pizza	Topping	Type
1	Mozzarella	Cheese
1	Pepperoni	Meat
1	Olives	Vegetable
2	Mozzarella	Cheese
2	Sausage	Meat
2	Peppers	Vegetable

Topping → Type

Pizza, Type → Topping

Is this in 3NF?

```
for (X→Y) in F
    Y is in X OR
    X is a superkey of R OR
    Y is part of a key in R
```

Wait, what just happened?

Redundancy is bad

Functional dependencies (FD)

- useful to find duplication

BCNF: No redundancy permitted!

- But may not be able to enforce FDs

3NF: Permits some duplication

- Can always decompose into 3NF

What's the point?

Improve our data design abilities
by understanding redundancy

We're going to need some theory

Closure of FDs infer new FDs from existing FDs

- armstrong's axioms

Minimal FD Set remove redundant FDs

Principled Decomposition

BCNF & 3NF

Closure of FDs

If I know
 Name → Bday and Bday → age
Then it implies
 Name → age

An FD f' is implied by set F if f' is true when F is true
F^+: the closure of F is all FDs implied by F

Can we construct this closure automatically? YES

Closure of FDs

Inference rules called Armstrong's Axioms
 Reflexivity if Y⊆X then X → Y
 Augmentation if X → Y then XZ → YZ for any Z
 Transitivity if X → Y & Y → Z then X → Z

These are sound and complete rules
 sound doesn't produce FDs not in the closure
 complete doesn't miss any FDs in the closure

Reflexivity: if Y⊆X then X → Y

A → A
A, B → A
X, Y, Z → Y, Z

The "trivial" rule:
 column always determines itself
 a set of columns determines any subset of those columns

Augmentation if X → Y then XZ → YZ for any Z

If: A → B

By reflexivity: C → C

… so … Stick them together? (informal)

A, C → B, C

Transitivity if X → Y & Y → Z then X → Z

Informal: apply them in sequence

X → Y: if you know (x, y) then X=x always implies Y=y
Y → Z: if you know (y, z) then Y=y always implies Z=z

Therefore, if you see X=x, you know Y=y;
and since you see y, you know Z=z

Closure of FDs

F = {A→B, B→C, CB→E}
Is A→E in the closure?

A → B	given
A → AB	augmentation A
A → BB	apply A→B
A → BC	apply B→C
BC → E	given
A → E	transitivity

We're going to need some theory

Closure of FDs
armstrong's axioms

Minimal FD Set

Principled Decomposition

BCNF & 3NF

Minimum Cover of FDs

Closures let us compare sets of FDs meaningfully
F1 = {A → B, A → C, A → BC}
F2 = {A → B, A → C}
F1 equivalent to F2

If there's a closure (a maximally expanded FD), there's a *minimal* FD. Let's find it

Minimum Cover of FDs

1. Turn FDs into *standard form*
   Split FDs so there is one attribute on the right side
2. Minimize left side of each FD
   For each FD, check if can delete a left attribute using another FD
   given ABC → D, B→C can reduce to AB→D, B→C
3. Delete redundant FDs
   check each remaining FD and see if it can be deleted
   e.g., in closure of the other FDs

2 must happen before 3!

Minimum Cover of FDs

A→B, ABC→E, EF→G, ACF→EG

Standard form (single attribute on right)
A→B, ABC→E, EF→G, ACF→E, ACF→G

Minimize left side
A→B, AC→E, EF→G, ACF→E, ACF→G
reason: AC→E + A→B implies ABC→E

Delete Redundant FDs
A→B, AC→E, EF→G, ~~ACF→E, ACF→G~~
ACF→E implied by AC→E, ACF→G implied by AC→E, EF→G

We're going to need some theory

Closure of FDs
armstrong's axioms

Minimal FD Set

Principled Decomposition

BCNF & 3NF

Decomposition

Eventually want to decompose R into $R_1 \ldots R_n$ wrt F

We've seen issues with decomposition.
Lost Joins: Can't recover R from $R_1 \ldots R_n$
Lost dependencies

Principled way of avoiding these?

Lossless Join Decomposition

join the decomposed tables to get *exactly the* original

e.g., decompose R into tables X, Y

$\pi_X(R) \bowtie \pi_Y(R) = R$

Lossless wrt F if and only if F^+ contains

$X \cap Y \rightarrow X$ or $Y \cap X \rightarrow Y$

intersection of X, Y is a key for one of them

Lossless Join Decomposition

Lossless wrt F if and only if F^+ contains

$X \cap Y \rightarrow X$ or $Y \cap X \rightarrow Y$

intersection of X, Y is a key for one of them

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

A	B	C
1	2	1
5	3	4
9	2	6

→

A	B
1	2
5	3
9	2

B	C
2	1
3	4
2	6

→

A	B	C
1	2	1
5	3	4
9	2	6
1	2	6
9	2	1

Lossy! $AB \cap BC = B$ doesn't determine anything

Lossless Join Decomposition

Lossless wrt F if and only if F^+ contains

$X \cap Y \rightarrow X$ or $Y \cap X \rightarrow Y$

intersection of X, Y is a key for one of them

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

A	B	C
1	2	1
5	3	4
9	2	6

→

A	B
1	2
5	3
9	2

A	C
1	1
5	4
9	6

→

A	B	C
1	2	1
5	3	4
9	2	6

OK

Dependency-preserving Decomposition

F_R = Projection of F onto R

FDs $X \rightarrow Y$ in F^+ s.t. X and Y attrs are in R

Subset of F that are "valid" for R

If R decompose to X,Y.

FDs that hold on X,Y equivalent to all FDs on R

$(F_X \cup F_Y)^+ = F^+$

Consider ABCD, C is key, $AB \rightarrow C$, $D \rightarrow A$

BCNF decomposition: BCD, DA

$AB \rightarrow C$ doesn't apply to either table!

We're going to need some theory

Closure of FDs

armstrong's axioms

Minimal FD Set

Principled Decomposition

BCNF & 3NF

BCNF

```
while BCNF is violated
   R with FDs F_R
   if X→Y violates BCNF
      turn R into R-Y & XY
```

Example

Branch, Customer, banker Name, Office
BCNO

Name -> Branch, Office N -> BO
Customer, Branch -> Name CB -> N

Example

Branch, Customer, banker Name, Office
BCNO

Name -> Branch, Office N -> BO
Customer, Branch -> Name CB -> N

CB is the key (determines everything)

BCNF

```
while BCNF is violated
   R with FDs F_R
   if X→Y violates BCNF
      turn R into R-Y & XY
```

BCNO BC → N, N → BO
NBO, CN using N → BO

uh oh, lost BC→N

3NF

F^{min} = minimal cover of F
Run BCNF using F^{min}
for X→Y in F^{min} not in projection onto $R_1 \ldots R_N$
create relation XY

BCNO BC → N, N → BO
NBO, CN using N → BO

3NF

F^{min} = minimal cover of F
Run BCNF using F^{min}
for X→Y in F^{min} not in projection onto $R_1 \ldots R_N$
create relation XY

BCNO BC → N, N → BO
NBO, CN using N → BO … oops create BCN
NBO, CN, BCN
NBO, BCN … BCN: BC is key; N → B violates BNCF

Summary

Normal Forms: BCNF and 3NF
FD closures: Armstrong's axioms
Proper Decomposition

Summary

Accidental redundancy is really really bad
Adding lots of joins can hurt performance

Can be at odds with each other
Normalization good starting point, relax as needed

People usually think in terms of entities and keys, usually ends up reasonable

What you should know

- Purpose of normalization
  - Anomalies
  - Decomposition problems
  - Functional dependencies & axioms
- 3NF & BCNF
  - properties
  - algorithm