CIS560

Obtaining a Good Database Design

1

Creating an Example Database Schema

- What do we need?
  - Order Number
  - Order Date
  - Customer Name
  - Billing address
  - Product Name
  - SKU
  - Product Category
  - Quantity
  - Unit Price

2

Good design provides:

- Data integrity
  - Maintains accuracy and consistency of the data
  - Data is recorded as intended
  - Data is retrieved as intended
  - In other words, helps avoid unintentional changes
- Easier maintenance of data
- Easier maintenance of code
  - SQL AND Application Code
- Better performance…usually

  Not always the case, but makes it more achievable.

Bad design…

- Lacks data integrity
- Decreases maintainability
- Performs poorly
- Introduces data anomalies
  - Redundancy

    With redundant data, what's the truth?
  - Update anomalies

    We have to update multiple places and may miss some.
  - Delete anomalies

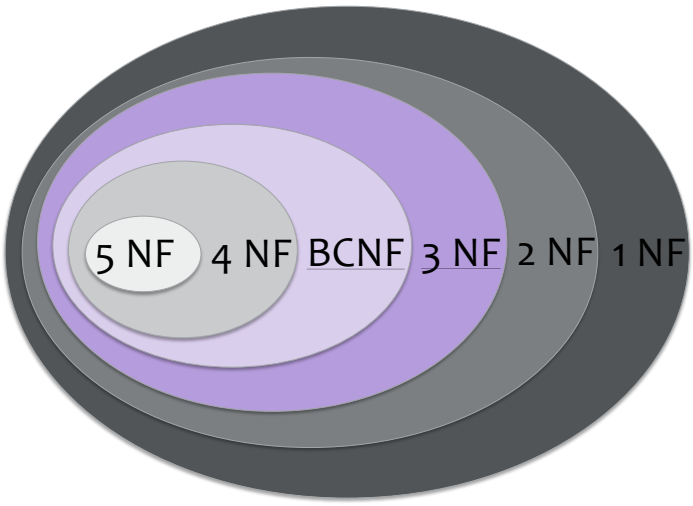
    We could unintentionally lose data.

How do we achieve good design?

- Database researchers developed normal forms.
- If a database is in one of the normal forms, then it is guaranteed to have certain properties.
- The most important, and most common, are:
  - Third Normal Form (3NF)
  - Boyce-Codd Normal Form (BCNF)
- 3NF and BCNF guarantee to avoid certain types of redundancy.

5

Normal Forms



6

Normal forms are defined using…

Keys
and
Functional Dependencies

7

Functional Dependencies

- They are a form of constraint.
- Finding them is a part of database design.
- Also used in normalizing the relations.
  1. Start with some relational schema.
  2. Find the functional dependencies.
  3. Use them to design a better schema.

8

Functional Dependencies Defined

- If two tuples agree on the attributes

$$A_1, A_2, \ldots, A_n$$

then they must agree on the attributes

$$B_1, B_2, \ldots, B_m$$

- Formally

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

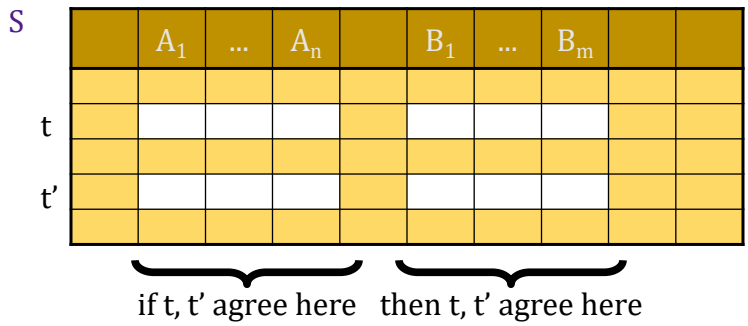
- Informally

If we know $A_1, A_2, \ldots, A_n$, then we know $B_1, B_2, \ldots, B_m$

When Does a Functional Dependency (FD) Hold?

Definition: $A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$ holds in relation S if:

$$\forall t, t' \in S, (t.A_1=t'.A_1 \wedge \ldots \wedge t.A_n=t'.A_n \Rightarrow t.B_1=t'.B_1 \wedge \ldots \wedge t.B_m=t'.B_m)$$



Example

A FD holds, or does not hold on an instance of a relation:

StudentID	Name	Phone	Dept
0045	Smith	1234	Math
3542	Mike	9876	CIS
1111	Smith	9876	CIS
9999	Mary	1234	Eng

StudentID → Name, Phone, Dept

Dept → Phone

Example

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

StudID	Name	Phone	Dept
0045	Smith	1234	Math
3542	Mike	9876 ←	CIS
1111	Smith	9876 ←	CIS
9999	Mary	1234	Eng

StudID → Name, Phone, Dept

Dept → Phone

Example

A FD holds, or does not hold on an instance of a relation:

StudID	Name	Phone	Dept
0045	Smith	1234	Math
3542	Mike	9876	CIS
1111	Smith	9876	CIS
9999	Mary	1234	Eng

StudID → Name, Phone, Dept Phone → Dept

Dept → Phone

Example

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

StudID	Name	Phone	Dept
0045	Smith	1234	Math
3542	Mike	9876	CIS
1111	Smith	9876	CIS
9999	Mary	1234	Eng

StudID → Name, Phone, Dept Phone → Dept

Dept → Phone

Example

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

StudID	Name	Phone	Dept
0045	Smith	1234	Math
3542	Mike	9876	CIS
1111	Smith	9876	CIS
9999	Mary	1234	Eng

StudID → Name, Phone, Dept

Phone → Dept

Dept → Phone

Example

FD's are constraints

- On some instances they hold
- On others they don't

name → color
category → store
color, category → price

Do all the above functional dependencies hold on this instance?

name	category	color	store	price
iPad	Gadget	Silver	Campus store	529
iPhone	Gadget	Silver	Campus store	429

Example

FD's are constraints

- On some instances they hold
- On others they don't

name → color
category → store
color, category → price

Do all the above functional dependencies hold on this instance?

name	category	color	store	price
iPad	Gadget	Silver	Campus store	529
iPhone	Gadget	Black	Campus store	429
iPad	Tablet	Silver	Best Buy	569

17

Class Exercise

If all these FDs hold:

name → color
category → store
color, category → price

Then this FD also holds:

name, category → price

Why?

18

Anomalies

Anomalies occur when "bad" FDs hold

- We know some of the FDs
- Need to find *all* FDs
- Then we can look for the bad ones

How do we find all Functional Dependencies?

Armstrong's Rules

Armstrong's Rules (1/3)

Splitting rule and Combining rule

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

Is equivalent to

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

21

Armstrong's Rules (2/3)

Trivial Rule

$A_1, A_2, \ldots, A_n \rightarrow A_i$

where i = 1, 2, ..., n

In other words:

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

22

Armstrong's Rules (3/3)

Transitive Closure Rule

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

and $B_1, B_2, \ldots, B_m \rightarrow C_1, C_2, \ldots, C_p$

then $A_1, A_2, \ldots, A_n \rightarrow C_1, C_2, \ldots, C_p$

23

Example

Start from the following FDs:

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

Infer the following FDs:

Inferred FD	Which Rule did we apply ?
4. name, category → name	Trivial Rule
5. name, category → color	Transitivity on 4, 1
6. name, category → category	Trivial Rule
7. name, category → color, category	Split/combine on 5, 6
8. name, category → price	Transitivity on 7, 3

24

THIS IS TOO HARD!

Let's see an easier way.

25

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 such that $A_1, \ldots, A_n \rightarrow B$

Example:

name → color
category → store
color, category → price

What are the closures?

$name^+$ = {name, color}
$\{name, category\}^+$ = {name, category, color, store, price}
$color^+$ = {color}

26

Closure Algorithm

$X=\{A_1, \ldots, A_n\}$.

Repeat until X doesn't change:

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:

name → color
category → store
color, category → price

$\{name, category\}^+ =$
{ name, category, color, store, price }

27

Closure Algorithm

$X=\{A_1, \ldots, A_n\}$.

Repeat until X doesn't change:

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:

name → color
category → store
color, category → price

$\{name, category\}^+ =$
{ name, category, color, store, price }

Hence: name, category → color, store, price

28

Class Exercise

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 }

29

Why Do We Need Closure?

- With closure we can find all Functional Dependencies
- You can check if X → A
  - Compute X^+
  - Check if A $\in$ X^+

30

Class Exercise

31

Using Closure to Infer ALL FDs

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

Functional Dependencies:

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

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

32