CSE 344: Section 10
Design Theory

March 8th, 2018

Today

Functional Dependencies (FD)

Boyce-Codd Normal Form (BCNF)

Annotated slides and notes posted for this material

Fair game on final but won't be worth a lot

Big Idea “Measure Twice, Cut Once”

E/R is mostly a visualization technique

Poor schemas can lead to inconsistency and performance inefficiencies

Updating a schema is expensive

Identify functional dependencies and normalize to **make well-behaved and fast databases the first time**

Motivating Example

We want to store information about **people**
(Name, SSN, PhoneNumber, City)

Known properties:

- Each person may have multiple phones
- Each person lives in only one city

Motivating Example

Is this a good representation of **people**?

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-123-4567	Seattle
Fred	123-45-6789	206-890-1234	Seattle
Joe	987-65-4321	626-246-8024	Los Angeles

Motivating Example

Is this a good representation of **people**?

Name	**SSN**	**PhoneNumber**	**City**
Fred	123-45-6789	206-123-4567	Seattle
Fred	123-45-6789	206-890-1234	Seattle
Joe	987-65-4321	626-246-8024	Los Angeles

No!

Motivating Example

Why is this a poor representation of **people**?

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-123-4567	Seattle
Fred	123-45-6789	206-890-1234	Seattle
Joe	987-65-4321	626-246-8024	Los Angeles

Anomalies:

-
-
-

Motivating Example

Why is this a poor representation of **people**?

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-123-4567	Seattle
Fred	123-45-6789	206-890-1234	Seattle
Joe	987-65-4321	626-246-8024	Los Angeles

Anomalies:

- Redundancy (data for Fred is duplicated)
-
-

Motivating Example

Why is this a poor representation of **people**?

Name	**SSN**	**PhoneNumber**	**City**
Fred	123-45-6789	206-123-4567	Seattle
Fred	123-45-6789	206-890-1234	Seattle
Joe	987-65-4321	626-246-8024	Los Angeles

Anomalies:

- Redundancy (data for Fred is duplicated)
- Slow Updates (what if Fred moved to Oahu?)
-

Motivating Example

Why is this a poor representation of **people**?

Name	**SSN**	**PhoneNumber**	**City**
Fred	123-45-6789	206-123-4567	Seattle
Fred	123-45-6789	206-890-1234	Seattle
Joe	987-65-4321	626-246-8024	Los Angeles

Anomalies:

- Redundancy (data for Fred is duplicated)
- Slow Updates (what if Fred moved to Oahu?)
- Zealous Deletion (what if Joe got rid of his phone?)

Motivating Example

How do we fix this?

Name	**<u>SSN</u>**	**<u>PhoneNumber</u>**	**City**
Fred	123-45-6789	206-123-4567	Seattle
Fred	123-45-6789	206-890-1234	Seattle
Joe	987-65-4321	626-246-8024	Los Angeles

Motivating Example

Normalization!

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-123-4567	Seattle
Fred	123-45-6789	206-890-1234	Seattle
Joe	987-65-4321	626-246-8024	Los Angeles

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

SSN	PhoneNumber
123-45-6789	206-123-4567
123-45-6789	206-890-1234
987-65-4321	626-246-8024

Motivating Example

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

SSN	PhoneNumber
123-45-6789	206-123-4567
123-45-6789	206-890-1234
987-65-4321	626-246-8024

Anomalies are gone!

- Minimal Redundancy
- Fast Updates
- Precise Deletion

Big Idea

Making a well-behaved and fast database can be done **systematically**!

Functional Dependencies (FD)

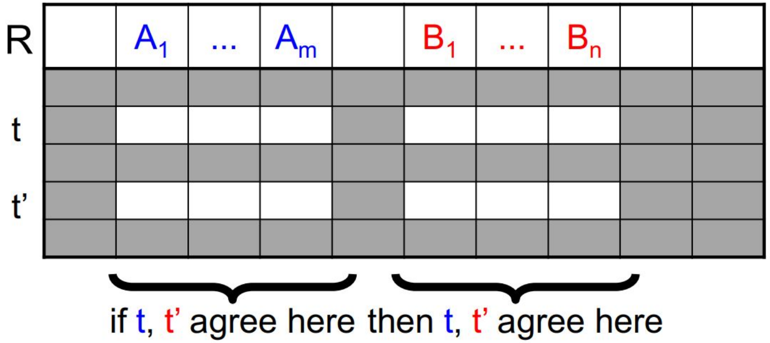
What is a Functional Dependency?

Formally:

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

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

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



What is a Functional Dependency?

Informally:

An FD holds when some attributes imply other attributes

What is a Functional Dependency?

SSN -> Name ?

SSN -> Name, City ?

SSN -> Name, City, PhoneNumber ?

Name	**<u>SSN</u>**	**<u>PhoneNumber</u>**	**City**
Fred	123-45-6789	206-123-4567	Seattle
Fred	123-45-6789	206-890-1234	Seattle
Joe	987-65-4321	626-246-8024	Los Angeles

What is a Functional Dependency?

SSN -> Name ?
Yes

SSN -> Name, City ?

SSN -> Name, City, PhoneNumber ?

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-123-4567	Seattle
Fred	123-45-6789	206-890-1234	Seattle
Joe	987-65-4321	626-246-8024	Los Angeles

What is a Functional Dependency?

SSN -> Name ?

Yes

SSN -> Name, City ?

Yes

SSN -> Name, City, PhoneNumber ?

Name	**<u>SSN</u>**	**<u>PhoneNumber</u>**	**City**
Fred	123-45-6789	206-123-4567	Seattle
Fred	123-45-6789	206-890-1234	Seattle
Joe	987-65-4321	626-246-8024	Los Angeles

What is a Functional Dependency?

SSN -> Name ?
Yes

SSN -> Name, City ?
Yes

SSN -> Name, City, PhoneNumber ?
No

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-123-4567	Seattle
Fred	123-45-6789	206-890-1234	Seattle
Joe	987-65-4321	626-246-8024	Los Angeles

What is a Functional Dependency?

SSN -> Name ?
Yes

SSN -> Name, City ?
Yes

Looks familiar?

SSN -> Name, City, PhoneNumber ?
No

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-123-4567	Seattle
Fred	123-45-6789	206-890-1234	Seattle
Joe	987-65-4321	626-246-8024	Los Angeles

What is a Functional Dependency?

SSN -> Name ?

Yes

SSN -> Name, City ?

Yes

Looks familiar?

SSN -> Name, City, PhoneNumber ?

No

Name	<u>SSN</u>	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Los Angeles

Finding FDs

Could be mapped from data… But usually, FDs should be established from prior knowledge about the data.

SSN -> Name

Name -> SSN

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-123-4567	Seattle
Fred	123-45-6789	206-890-1234	Seattle
Joe	987-65-4321	626-246-8024	Los Angeles

Finding FDs

Could be mapped from data... But usually, FDs should be established from prior knowledge about the data.

SSN -> Name ✔

Name -> SSN true for now...

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-123-4567	Seattle
Fred	123-45-6789	206-890-1234	Seattle
Joe	987-65-4321	626-246-8024	Los Angeles

Closure Algorithm

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.

Goal: We want everything that an attribute/set of attributes determine

Observation:

If we have A -> B and B -> C, then A -> C

Closure Algorithm

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.

Goal: We want everything that an attribute/set of attributes determine

Observation:

If we have A -> B and B -> C, then A -> C

So really, A -> B and C

Closure Algorithm

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.

Goal: We want everything that an attribute/set of attributes determine

Observation:

If we have A -> B and B -> C, then A -> C

So really, A -> B and C

Formal notation is $\{A\}^+ = \{A, B, C\}$

Closure Algorithm

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.

Goal: We want everything that an attribute/set of attributes determine

Observation:

If we have A -> B and B -> C, then A -> C

So really, A -> B and C

Formal notation is $\{A\}^+ = \{A, B, C\}$

Since the closure of A is all attributes, A is a key

Keys

We call an attribute that determines all other attributes in a schema to be a **superkey**.

If it is the smallest set of attributes (in terms of cardinality) that does this we call that set a **minimal key** or just **key**

Anomalies

X -> Y in your table schema implies an anomaly UNLESS X is a (super)key

We deal with this by normalizing the schema (i.e. ripping apart tables until these anomalies are gone)

Boyce-Codd Normal Form (BCNF)

What is a “Normal Form”?

Goal of normal forms is to promote consistency, speed, ease of use, etc.

1st Normal Form: Tables are flat

2nd Normal Form: Obsolete

BCNF: No bad FDs

3rd Normal Form: See textbook for more details

Non-trivial dependency: X -> X

What is BCNF?

Definition. A relation R is in BCNF if:

Whenever $X \rightarrow B$ is a non-trivial dependency, then X is a superkey.

=

Definition. A relation R is in BCNF if:

$\forall$ X, either $X^+ = X$ or $X^+ =$ [all attributes]

Conversion to BCNF

Not covered this quarter:

Lossless decomposition

Chase algorithm

High level database design software implements these algorithms to optimize your schema automatically.

Recap

WTH Just Happened

Functional dependencies give us hints about the possible keys for relations

If we can establish useful FDs, we can systematically create well-formed schemas.

Practical Tips

Normalization is great for promoting consistency about current states

Fully normalized data can be hindering (think about joins). Denormalizing can bring back redundancy but improve performance in some cases.