

Data Quality

Eugene Wu

Data Quality

Prevention

- Normalization prevents inconsistencies
- Constraints prevent invalid values

Detection

- Incorrect/anomalous values
- Deduplication

Steps for a New Application

Requirements

what are you going to build?

Conceptual Database Design

pen-and-pencil description

Logical Design

formal database schema

Schema Refinement:

fix potential problems, normalization

Normalization

Physical Database Design

use sample of queries to optimize for speed/storage

App/Security Design

prevent security problems

A Relational Model of Data for Large Shared Data Banks

E. F. CODD

IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report

Redundancy = Bad

What was our solution?

- Break database into small relations

- Perform “good” ER modeling and SQL translation

- Kind of ... adhoc

Is there a systematic approach?

Redundancy = Bad

<u>sid</u>	name	address	<u>hobby</u>	cost
1	Eugene	amsterdam	trucks	\$\$
1	Eugene	amsterdam	cheese	\$
2	Bob	40th	paint	\$\$\$
3	Bob	40th	cheese	\$
4	Shaq	florida	swimming	\$

people have names and addrs

hobbies have costs

people many-to-many with hobbies

What's primary key? *sid?* *sid + hobby?*

Update/insert/delete anomalies. Wastes space

Anomalies (Inconsistencies)

Update Anomaly

change one address, need to change all

Insert Anomaly

add person without hobby?
not allowed? dummy hobby?

Delete Anomaly

if delete a hobby. Delete the person? Set hobby to null?

Theory Can Fix This!

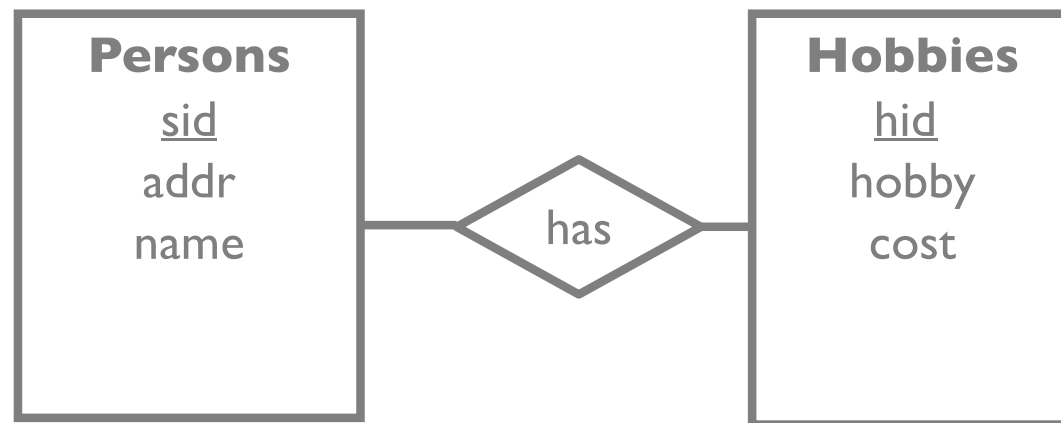
A Possible Approach

Persons	
<u>sid</u>	
addr	
name	
hobby	
cost	

data(sid, addr, name, hobby, cost)

A Possible Approach

ER diagram was a heuristic



We have decomposed example table into:

person(sid, **addr**, **name**)

hobby(name, **cost**)

personhobby(hobbyname, sid)

WHY is this a good decomposition??

A Possible Approach

What if decompose into:

person(sid, name, addr, cost)

personhobby(sid, hobbyname)

<u>sid</u>	name	addr	cost
1	Eugene	amsterdam	\$\$
1	Eugene	amsterdam	\$
2	Bob	40th	\$\$\$
3	Bob	40th	\$
4	Shaq	florida	\$

<u>sid</u>	hobby
1	trucks
1	cheese
2	paint
3	cheese
4	swimming

but... which cost goes with which hobby?

lost information: *lossy decomposition*

Decomposition

Replace schema R with 2+ smaller schemas that

1. each contain subset of attrs in R
2. together include all attrs in R

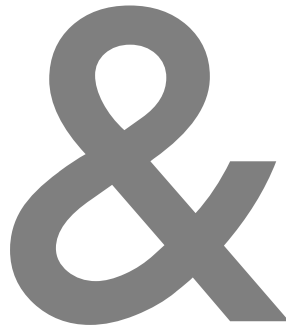
ABCD replaced with AB, BCD or AB, BC, CD

Not free – may introduce problems!

1. lossy-join: not able to recover R from smaller relations
2. non-dependency-preserving: constraints on R cannot be enforced y only looking at an individual decomposed relation
3. performance: additional joins, may affect performance

Can we systematically
decompose our relation to

prevent
decomposition
problems



remove
redundancy?

Functional Dependencies (FD)

sid	name	address	hobby	cost
1	Eugene	amsterdam	trucks	\$\$
1	Eugene	amsterdam	cheese	\$
2	Bob	40th	paint	\$\$\$
3	Bob	40th	cheese	\$
4	Shaq	florida	swimming	\$

sid sufficient to identify name and addr, but not hobby

e.g., exists a function $f(\text{sid}) \rightarrow \text{name, addr}$

sid \rightarrow name, addr is a **functional dependency**

“sid determines name, addr”

“name, addr are functionally dependent on sid”

“if 2 records have the same sid, their name and addr are the same”

Functional Dependencies (FD)

$$X \rightarrow Y$$

holds on R

if $t_1.X = t_2.X$ then $t_1.Y = t_2.Y$

where X, Y are subsets of attrs in R

Examples of FDs in person-hobbies table

$\text{sid, hobby} \rightarrow \text{name, address cost}$

$\text{hobby} \rightarrow \text{cost}$

$\text{sid} \rightarrow \text{name, address}$

Redundancy depends on FDs

consider $R(ABC)$

no FDs: no redundancy

if $A \rightarrow B$: tuples with same A value means B is duplicated!



Fun Facts

Functional Dependency is an integrity constraint
statement about all instances of relation

Generalizes key constraints

if K is candidate key of R , then $K \rightarrow R$

Given FDs, simple definition of redundancy

when left side of FD is not table key

Where do FDs come from?

thinking really hard aka application semantics

can't stare at database to derive (same as ICs)

Fun Facts

Functional Dependency is an integrity constraint
statement about all instances of relation

Generalizes key constraints

if K is candidate key of R , then $K \rightarrow R$

Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms

Thorsten Papenbrock²

Jens Ehrlich¹

Jannik Marten¹

Tommy Neubert¹

Jan-Peer Rudolph¹

Martin Schönberg¹

Jakob Zwiener¹

Felix Naumann²

¹ firstname.lastname@student.hpi.uni-potsdam.de

² firstname.lastname@hpi.de

Hasso-Plattner-Institut, Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany

We're going to need some theory

Closure of FDs

armstrong's axioms

Principled Decomposition

BCNF

Closure of FDs

Given

$\text{Name} \rightarrow \text{Bday}$ and $\text{Bday} \rightarrow \text{age}$

We know

$\text{Name} \rightarrow \text{age}$

f' is implied by set F if f' is true when F is true

F^+ is the **closure** of F . Defined as all FDs implied by F

Can we construct this closure automatically? YES

Closure of FDs

Inference rules called Armstrong's Axioms

Reflexivity if $Y \subseteq X$ then $X \rightarrow Y$

Augmentation if $X \rightarrow Y$ then $XZ \rightarrow YZ$ for any Z

Transitivity if $X \rightarrow Y$ & $Y \rightarrow Z$ then $X \rightarrow Z$

These are **sound** and **complete** rules

sound doesn't produce FDs not in the closure

complete doesn't miss any FDs in the closure

Closure of FDs

Can we compute the closure? YES. slowly
expensive. exponential in # attributes

Can we *check* if $X \rightarrow Y$ is in the closure of F?

$X^+ = \text{attribute closure}$ of X (expand X using axioms & FDs)

check if Y is implied in the attribute closure

Closure of FDs

Approach 1: Infer new functional dependencies

$F = \{A \rightarrow B, B \rightarrow C, CB \rightarrow E\}$

Is $A \rightarrow E$ in the closure?

$A \rightarrow B$

given

$A \rightarrow AB$

augmentation A

$A \rightarrow BB$

apply $A \rightarrow B$ (transitivity)

$A \rightarrow BC$

apply $B \rightarrow C$ (transitivity)

$A \rightarrow E$

apply $BC \rightarrow E$ (transitivity)

Closure of FDs

Approach 2: Infer new attributes starting from left hand side of “goal” functional dependency.

$F = \{A \rightarrow B, B \rightarrow C, CB \rightarrow E\}$

Is $A \rightarrow E$ in the closure?

A	Left hand side of “goal”
$\rightarrow A$	reflexivity
$\rightarrow B$	apply $A \rightarrow B$ (transitivity)
$\rightarrow C$	apply $B \rightarrow C$ (transitivity)
$\rightarrow E$	apply $BC \rightarrow E$ (transitivity)

We're going to need some theory

Closure of FDs

armstrong's axioms

Principled Decomposition

BCNF

Decomposition

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

We've seen issues with decomposition.

Lossy Joins: Can't recover R from $R_1 \dots R_n$

Lost dependencies

Principled way of avoiding these?

Lossless Join Decomposition

Let's say relation R is decomposed into relations X,Y

Join the decomposed tables to get *exactly the* original

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

Lossless wrt F if and only if F^+ contains

$$X \cap Y \rightarrow X \text{ or } Y \cap X \rightarrow Y$$

$X \cap Y$ is a key for X or Y or both

Lossless Join Decomposition

Lossless wrt F if and only if F^+ contains

$$X \cap Y \rightarrow X \text{ or } Y \cap X \rightarrow Y$$

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

$$X \cap Y \rightarrow X \text{ or } Y \cap X \rightarrow Y$$

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

Subset of F^+ that are “valid” for R

FDs $X \rightarrow Y$ in F^+ where X and Y attrs are in R

If R decomposed into relations X and Y .

FDs that hold on X, Y are 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. Not dependency preserving.

We're going to need some theory

Closure of FDs

armstrong's axioms

Principled Decomposition

BCNF

BCNF

Relation R in BCNF has *no redundancy* wrt FDs
(FDs are all key constraints)

F: set of functional dependencies over relation R

for $(X \rightarrow Y)$ in F^+
Y is in X *or*
X is a superkey of R

Is this in BCNF?

$\text{sid} \rightarrow \text{name}$

sid	hobby	name
x	y ₁	z
x	y ₂	?

BCNF

Relation R in BCNF has *no redundancy* wrt FDs
(FDs are all key constraints)

F: set of functional dependencies over relation R

for $(X \rightarrow Y)$ in F^+

Y is in X *or*

X is a superkey of R (X's closure wrt F^+ contains R)

Functional Dependencies

$SH \rightarrow NAC$ (sid, hobby \rightarrow name, addr, cost)

$H \rightarrow C$

$S \rightarrow NA$

What's in BCNF?

SHNAC NO

SNA, SHC NO

SNA, HC, SH YES

BCNF

Relation R in BCNF has *no redundancy* wrt FDs
(FDs are all key constraints)

F: set of functional dependencies over relation R

for $(X \rightarrow Y)$ in F^+

Y is in X *or*

X is a superkey of R (X's closure wrt F^+ contains R)

Remember that F^+ is *all* possible FDs valid over R!

Given $A \rightarrow B$, $B \rightarrow C$, then $A \rightarrow C$ is in F and should be checked

BCNF

Suppose we have

Client, Office \rightarrow Account

Account \rightarrow Office

What's in BCNF?

R(Account, Client, Office)

R(Account, Office) R(Client, Account)

Where did $CO \rightarrow A$ go? *Lost a Functional Dependency*

BCNF does not preserve FDs

BCNF Examples

Functional Deps	Decomposition	In BCNF?
$A \rightarrow B, B \rightarrow C,$	ABC	NO. $B \rightarrow C$ violated
$A \rightarrow B, B \rightarrow C, C \rightarrow A$	ABC	YES
$AB \rightarrow C, CD \rightarrow E, C \rightarrow A$	ABCDE	NO. All FDs violated
	ABC, CDE	NO. $C \rightarrow A$ violates
	CA, AB, CDE	YES
$AB \rightarrow C, CD \rightarrow E, C \rightarrow AB$	ABC, CDE	YES

BCNF Decomposition

while BCNF is violated

R with FDs F_R

if an FD $X \rightarrow y$ violates BCNF

turn R into $R - y$ & Xy

Check FDS in R's projection

y is single attr

ABCDE

key A, $BC \rightarrow A$, $D \rightarrow B$, $C \rightarrow D$

Consider $D \rightarrow B$

Decompose into DB, ACDE

DB now in BCNF

Consider $C \rightarrow D$

B not in ACDE, so $BC \rightarrow A$ and $D \rightarrow B$ not in projection

Decompose into ACE, CD

Consider key A

A determines ACE, no other FDs on ACE to check

DB, CD, ACE

BCNF Decomposition Example

$A \rightarrow BC, C \rightarrow D, BD \rightarrow E$

$A \rightarrow B, A \rightarrow C, C \rightarrow D, BD \rightarrow E$

ABCDE is starting relation

Ensure right side has 1 attr

Consider $A \rightarrow B$

A determines BCDE. OK

Consider $C \rightarrow D$

C doesn't determine E

CD, ABCE

Consider $A \rightarrow B$

A determines BC, but not E

CD, AB, ACE

Consider $A \rightarrow C$

A doesn't determine E

CD, AB, AC, AE

OK

BCNF Decomposition Example

$A \rightarrow BC, C \rightarrow D, BD \rightarrow E$

$A \rightarrow B, A \rightarrow C, C \rightarrow D, BD \rightarrow E$

ABCDE is starting relation

Ensure right side has 1 attr

Consider $BD \rightarrow E$

BD doesn't determine A

BDE, ABCD

Consider $C \rightarrow D$

C doesn't determine A

BDE, CD, CAB

Consider $A \rightarrow B$

A determines BC! In BCNF

Done

BCNF decomposition is not unique. Depends on order of FDs

Other Normal Forms

Two different criterias for decomposing a relation

Boyce Codd Normal Form (BCNF)

No redundancy, may lose dependencies

3rd Normal Form (3NF)

May have redundancy, no decomposition problems

Common

1st, 2nd, 4th, ... normal forms

Common Data Quality Issues (incomplete list)

Value Errors

- Finding outliers
- Robustness to outliers
- Deal with outliers/value errors (Imputation)

Entity Resolution

Value Errors

Values in an attribute that appear wrong

- missing
- placeholder values e.g., 0, -9999, “NA”, “Empty”
- “anomalous” values (outliers)

Why are outliers bad?

- causes estimates (AVG) to be arbitrarily wrong
- $\text{AVG}(2, 2, 2, 2, 5) = 2.6$
- $\text{AVG}(2, 2, 2, 2, 100) = 21.6$
- $\text{AVG}(2, 2, 2, 2, 1000) = 201.6$

Finding Outliers

How to detect (numeric) outliers?

- Fundamentally a modeling problem.
- What's a “normal” value? Assume a distribution

Example using age

- Assume a normal distribution
- Fit age values to distribution (compute avg, stddev)
- Low-probability ages (outside of $\text{mean} \pm \text{std}$) labeled outliers

Robustness to Outliers

Can the query be resilient to outliers?

Breakdown point:

- % of incorrect values before statistic is arbitrarily wrong
- AVG: one value
- Median: 50%

1, 2, 3, 4, 5, 6, 7, 8, 100

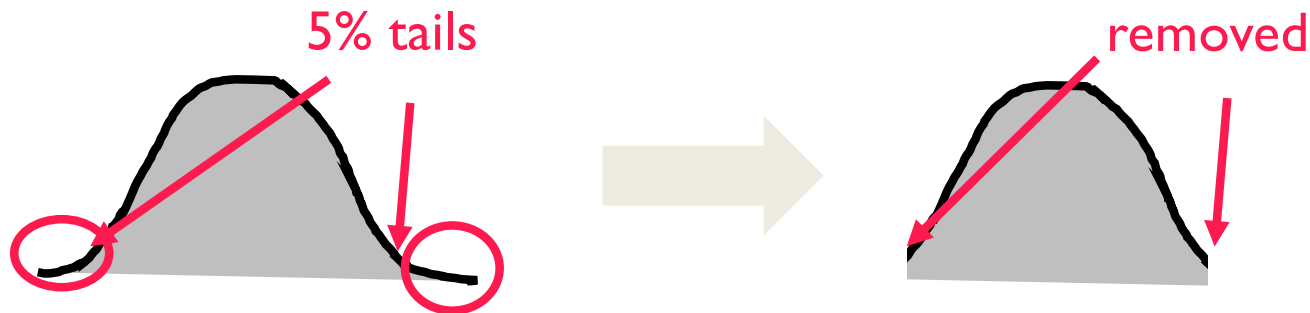
AVG 15.1

Median 5 Sort and return middle value

Deal With Outliers

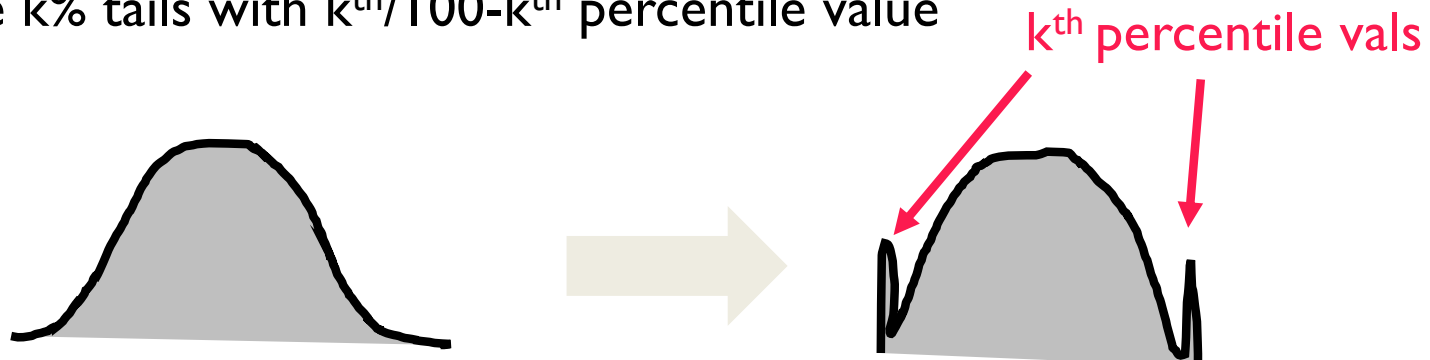
Trimming 5%

- remove top and bottom 5% percentile (the “tails”)
- p_5, p_{95}



Winsorize

- Replace extreme values with closest value
- Replace $k\%$ tails with $k^{\text{th}}/100$ - k^{th} percentile value



Imputing Missing Value

Fill in with “likely” values

Why?

- Some apps don't accept nulls
- Lead to bias/errors

How to define “likely”?

- default values (mean/median, user-defined val)
- interpolate (if ordered data)
- build a model and predict

Entity Resolution

Same entity, different “physical representations”

Example: text attributes due to e.g., misspellings

- New York, NYC, New York City, NY City, ...
- Different forms of string similarity

```
SELECT *  
FROM cities c1, cities c2  
WHERE levenshtein(c1.name, c2.name) < 3
```

Entity Resolution

Example: different rows refer to the same entity.

- Seller 1: (123, “iphone max 12Pro,” \$1000)
- Seller 2: (00123, “IPhone12 Pro MAX”, “85722.40 Rubles”)
- Fundamentally classification problem. `is_same(row1, row2) → T/F`

```
SELECT row1, row2
FROM data as d1, data as d2
WHERE is_same(d1.*, d2.*)
```

Entity Resolution

Very hard problem! No “Solution”.

10000 records. 100,000,000 possible matches!

3 main steps

1. **Blocking**: group data into subsets (may overlap)
2. **Matching**: within each group:
 1. choose distance function between rows
 2. cluster rows using distance function.

Hope that same entities in same cluster, and
different entities in different clusters

Entity Resolution

Blocking:

- Extract grouping features/attributes A
- Form groups based on same/similar A values
- Each group is a “block”

Simple approach: group by an attribute

Example for strings: group by shared n-gram

2-gram for “eugene”: eu, ug, ge, en, ne

All words containing “eu” in one block

Choose n via trial-and-error

Entity Resolution: Matching

Distance metrics

Single attribute

- string similarity functions
- synonym dictionary: fast = quick
- knowledge-base: *mandarin* closer to *orange* than banana
- embeddings

Multi-attribute:

- linear combination of attribute distances
(4111, “intro to db”), (COMS4111, “Introducton to DB”),
(4112, “db impl”)
- train a model if you have labels

Summary

Normalization

- Functional dependencies & redundancy
- FD closures: Armstrong's axioms
- Proper Decomposition into BCNF

Data Quality

- Value errors.
- Entity resolution

Summary

Trade-off

- Accidental redundancy is really really bad
- But adding lots of joins can hurt performance

In practice:

- List FDs
- normalize,
- decide which decompositions to skip as needed

What you should know

Purpose of normalization

- Anomalies
- Decomposition problems
- Functional dependencies & axioms

BCNF

- properties
- algorithm

Data Quality (level of lecture)

- Value errors: How to detect, avoid, fix
- Entity resolution: What it is, why it's hard, overall steps

Why The Term “Normalize Relations”?

Designing a database involves a process called normalization where the data model is broken-down according its smallest granularity. Invented by Ted Codd and Chris Date, the term normalization was borrowed from President Richard Nixon normalizing relations with China in the 1970's. Codd stated that if Nixon could normalize relations, then so could the relational model.

Exercises

w4l1l.github.io/fd

Functional Dependency Problem Generator

We have generated 99 random functional dependency problems for you to have practice with. You can press ← or → on the keyboard to go to the previous or next problem.

Click to Toggle Answer



Go to random question



Problem 0 out of 100

Info

Relation	ABCDEFGH
Functional Deps	FD -> HE FE -> DB C -> FED GB -> E
Is BCNF?	
Is 3NF?	

Minimal FDs

List the minimal closure for the functional dependencies

Decomposition

BCNF using FDs

List the BCNF decomposition using the provided functional dependencies:

BCNF using Minimal Cover

List the BCNF decomposition using the minimal closure of the functional deps (this is just to give you more decomposition exercises):

3NF

List the 3NF decomposition: