

**L17**

**Normalization is a Good Idea**

# Administrivia

Midterms returned today!

Project Part 3 due **next Tuesday** March 29

Part 3 demos: March 28-April 1

Mentor should have contacted you

HW3: Available now; due April 5

# 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

# **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 is no good

Update/insert/delete anomalies. Wastes space

<u>sid</u>	name	address	hobby	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?

# Anomalies (Inconsistencies)

## Update Anomaly

change one address, need to change all

## Insert Anomaly

add person without hobby?  
not allowed? dummy hobby?

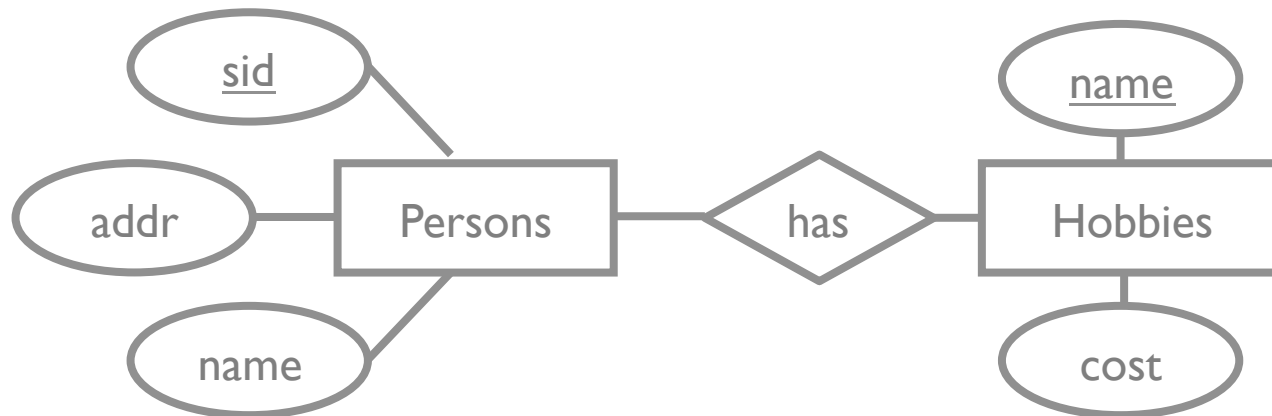
## Delete Anomaly

if delete a hobby. Delete the person?

**Theory Can Fix This!**

# A Possible Approach

ER diagram was a heuristic



We have decomposed example table into:

person(sid, addr, name)

hobby(name, cost)

personhobby(hobbyname, sid)

# A Possible Approach

What if decompose into:

person(sid, name, address, cost)

personhobby(sid, hobbyname)

<u>sid</u>	name	address	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

## Desirable properties

1. Lossless join: able to recover R from smaller relations
2. Dependency preserving: enforce constraints on R by only enforcing constraints on smaller schemas (no joins)

# Decomposition is a trade-off

## Advantages:

- Eliminates possibility of data getting “out of sync”
- Make changes in one place that apply everywhere
- Access a sub-section of the data for some queries

## Disadvantages

- If always need all data together, joins may be slower
- Less flexible because there is no redundancy

How can we systematically  
decompose relations to 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

$sid \rightarrow name, address$

$hobby \rightarrow cost$

$sid, hobby \rightarrow name, address cost$

$X \rightarrow Y$  is a functional dependency

$$Y = f(X)$$

# 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 (like ICs)

Like a Mathematics conjecture:

one counter example can disprove, but examples can't prove  
there are no example in the universe



# Where do FDs come from?

thinking really hard aka application semantics  
can't stare at database to derive (like ICs)

Like a Mathematics conjecture:

## **Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms**

Thorsten Papenbrock<sup>2</sup>

Tommy Neubert<sup>1</sup>

Jens Ehrlich<sup>1</sup>

Jan-Peer Rudolph<sup>1</sup>

Jannik Marten<sup>1</sup>

Martin Schönberg<sup>1</sup>

Jakob Zwiener<sup>1</sup>

Felix Naumann<sup>2</sup>

<sup>1</sup> firstname.lastname@student.hpi.uni-potsdam.de

<sup>2</sup> firstname.lastname@hpi.de

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

# Normal Forms

Criteria met by a relation R wrt functional dependencies

Boyce Codd Normal Form (BCNF)

No redundancy, may lose dependencies

Third Normal Form (3NF)

May have redundancy, no decomposition problems

Redundancy depends on FDs

consider R(ABC)

no FDs: no redundancy

if  $A \rightarrow B$ : B is duplicated if there are multiple copies of A

# BCNF

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

F: set of functional dependencies over relation R

X: Subset of attributes of R

A: One attribute of R

for  $(X \rightarrow A)$  in F

A is in X OR

X is a superkey of R

sid, hobby, name, addr, cost

$H \rightarrow C$  (hobby  $\rightarrow$  cost)

$S \rightarrow NA$

What's in BCNF?

for  $(X \rightarrow A)$  in F

A is in X OR

X is a superkey of R

SHNAC NO

SNA, SHC NO

SNA, HC, SH YES

# BCNF

Relation R in BCNF has *no redundancy* wrt FDs  
(only FDs are 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

No as repeated rows of x, and different y\_1 and y\_2

Is this in BCNF?

sid  $\rightarrow$  name

sid	hobby	name
x	y <sub>1</sub>	z
x	y <sub>2</sub>	?

# 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)

If have (Pizza, Topping, Type), then can have additional row of cheese

# Pizza: Dependencies?

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

Topping  $\rightarrow$  Type

Pizza, Type  $\rightarrow$  Topping

Is this in BCNF?

# Pizza BCNF

Topping  $\rightarrow$  Type

Pizza, Type  $\rightarrow$  Topping

for  $(X \rightarrow A)$  in  $F$

A is in X OR

X is a superkey of R



# Pizza BCNF

**Topping  $\rightarrow$  Type**

Pizza, Type  $\rightarrow$  Topping

for  $(X \rightarrow A)$  in  $F$

~~A is in X OR~~

~~X is a superkey of R~~

# 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  $\rightarrow$  Type

Pizza, Type  $\rightarrow$  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

# 3<sup>rd</sup> Normal Form (3NF)

Relax BCNF (e.g.,  $BCNF \subseteq 3NF$ )

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

# 3<sup>rd</sup> Normal Form (3NF)

Relax BCNF (e.g.,  $BCNF \subseteq 3NF$ )

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 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  $\rightarrow$  Type

Pizza, Type  $\rightarrow$  Topping

Is this in 3NF?

# Pizza: Dependencies?

Topping  $\rightarrow$  Type

Pizza, Type  $\rightarrow$  Topping

for  $(X \rightarrow Y)$  in  $F$   
Y is in X OR  
X is a superkey of R OR  
Y is part of a key in R

Victory! This is in 3<sup>rd</sup> Normal Form  
(Topping determines part of a key)

# 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