

Administrivia

Project Part 3 is out
HW3 is out – lots of lead time

Scribe notes
Examples of dbapis today
OH Wednesday after class – go over setting up a webserver

L16

Normalization is a Good Idea

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

Information Retrieval

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

What do we think about redundancy?

NOT GOOD

Redundancy is no good

Update/insert/delete anomalies. Wastes space

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	\$

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?

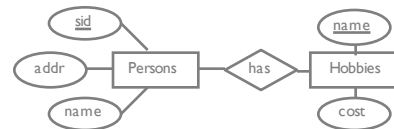
Delete Anomaly

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 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 $f(sid) \rightarrow \text{name, addr}$

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 (why?)

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 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.-Helmut-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

Is this in BCNF?

sid \rightarrow name

sid	hobby	name
x	y ₁	z
x	y ₂	?

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 \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