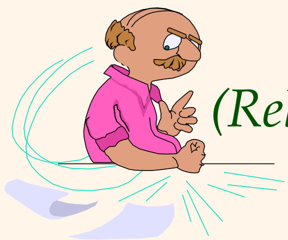
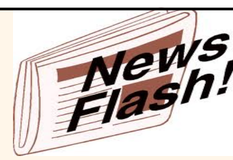
Introduction to Data Management



Lecture #7 (Relational Design Theory)

Instructor: Mike Carey
mjcarey@ics.uci.edu

Announcements

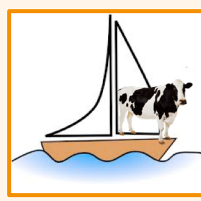


- Thank you, Shiva, for covering last week!
- HW#2 is underway...! (From *HW #1 solution*)
  - Use your best judgement on data types and NULLs
  - Capture all the capturable ICs (PK/FK/UNIQUE)
- A few logistical notes (office hours, discussions)
  - ~450 ≈ Hybrid F2F/online (Piazza) class (😐)
  - Definitely bring your questions to discussion section meetings (including the just-finished HW solution)
- Today's plan:
  - Relational DB design theory
  - *Disclaimer*: Not the most exciting part of CS122A… ☺

So, Given a Relational Schema...

- How do I know if my relational schema is a "good" logical database design or not?
  - What might make it "not good"?
  - How can I fix it, if indeed it's "not good"?
  - How "good" is it, after I've fixed it?
- Note that your relational schema might have come from one of several places
  - You started from an E-R model (but maybe that model was "wrong" or incomplete in some way?)
  - You went straight to relational in the first place
  - It's not your schema – you inherited it! ☺

Ex: Wisconsin Sailing Club

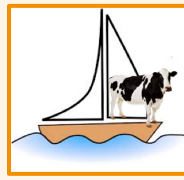


Proposed bad schema design:

sid	sname	rating	age	date	bid	bname	color
22	Dustin	7	45.0	10/10/98	101	Interlake	blue
22	Dustin	7	45.0	10/10/98	102	Interlake	red
22	Dustin	7	45.0	10/8/98	103	Clipper	green
22	Dustin	7	45.0	10/7/98	104	Marine	red
31	Lubber	8	55.5	11/10/98	102	Interlake	red
31	Lubber	8	55.5	11/6/98	103	Clipper	green
31	Lubber	8	55.5	11/12/98	104	Marine	red
...	...	...	...	...	...	...	...

Review: **Bad** design due to redundancy and its problems.

Ex: Wisconsin Sailing Club



Proposed good schema design:

sid	sname	rating	age
22	Dustin	7	
31	Lubber	8	
...	...	...	

sid	bid	date
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
...	...	...

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Review: **Good** design due to elimination of redundancy.

The Evils of Redundancy (or: The Evils of Redundancy)

- *Redundancy* is at the root of several problems associated with relational schemas:
  - Redundant storage (space)
  - Insert/delete/update anomalies

A good rule to follow:
"One fact, one place!"

- *Functional dependencies* can help in identifying problem schemas and suggesting refinements.
- Main refinement technique: *decomposition*, e.g., replace R(ABCD) with R1(AB) + R2(BCD)
- Decomposition should be used judiciously: 明智的
  - Is there reason to decompose a relation?
  - Does the decomposition cause any problems?

Functional Dependencies (FDs)

可以多指向一
可以一指向一
不可以一指向多

- A <u>functional dependency</u> X → Y holds over relation R if, for every allowable instance *r* of R:
  - For *t1* and *t2* in *r*, *t1*.X = *t2*.X implies *t1*.Y = *t2*.Y
  - I.e., *given two tuples in r, if the X values agree, then their Y values must also agree*. (X and Y can be *sets* of attributes.)
- An FD is a statement about *all* allowable relations.
  - Identified based on *application semantics* (similar to E-R).
  - Given some instance *r1* of R, we can check to see if it violates some FD *f*, but we cannot tell if *f* holds over R!
- Saying K is a candidate key for R means that K→ R
  - Note: K → R alone does not require K to be *minimal*! If K is minimal, then K is a candidate key (else it's a super key).

Example: Constraints on an Entity Set

- Suppose you're given a relation called HourlyEmps:
  - HourlyEmps (<u>*ssn*</u>, *name*, *lot*, *rating*, *hrly_wages*, *hrs_worked*)
- <u>*Notation*</u>: Let's denote this relation schema by simply listing the attributes: SNLRWH
  - This is really the *set* of attributes {S,N,L,R,W,H}.
  - Sometimes, we will refer to *all* attributes of a relation by using the relation name (e.g., HourlyEmps for SNLRWH).
- Suppose we also have some FDs on HourlyEmps:
  - *ssn* is the key: S → SNLRWH
  - *rating* determines *hrly_wages*: R → W

因为S -> SNLRWH
R->W

所以把那个大table的R指向的拿出来，然后大table就可以没有这个，但是要留R，小table就R和W因为R->W

Example (Cont'd.)

- Problems due to R → W :
  - *Update anomaly*: What if we change W in just the 1st tuple of SNLRWH?
  - *Insertion anomaly*: What if we want to insert a new employee and don't know the proper hourly wage for his or her rating?
  - *Deletion anomaly*: If we delete all employees with rating 5, we lose the stored information about the wage for rating 5!

How about two smaller tables?

Wages

R	W
8	10
5	7

+

HourlyEmps2

S	N	L	R	H
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Reasoning About FDs

- Given some FDs, we can usually infer additional FDs:
  - *ssn* → *did*, *did* → *lot* implies *ssn* → *lot*
  - (Translation: Matching *ssns* imply matching *lots*.)
- An FD *f* is *implied by* a set of FDs *F* if *f* holds whenever all FDs in *F* hold. F+指的是ABCDEF全部
  - F^+ = *closure of F* is the set of *all* FDs that are implied by *F*. F+是指F可以控制的所以FDs
- **Armstrong's Axioms** (X, Y, Z are sets of attributes):
  - *Reflexivity*: If $X \subseteq Y$, then $Y \rightarrow X$
  - *Augmentation*: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
  - *Transitivity*: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- These are *sound* and *complete* inference rules for FDs!

Armstrong's Axioms: Examples

pno	name	title	state	zip
1	Sandy	Professor	CA	92697
2	Joe	Jim Gray Professor	CA	94720
3	Anhai	Professor	WI	53706
4	Alex	Associate Professor	CA	92697

- *Reflexivity*: If X⊆ Y then Y→X:
  - **zip** ⊆ (**zip**, name), so (**zip**, name) → **zip.**

因为zip是super key里面的某一个元素

- *Augmentation*: If X→Y then XZ→YZ for any Z:
  - zip → state, so (zip, **title**) → (state, **title**).
- *Transitivity*: If X→Y and Y→Z then X→Z:
  - **pno** → zip and zip → **state**, so **pno** → **state.**

candidate key是最少key可以管所有的
primary key是大table的keys
primary key只有一个，candidate可以是多个
primary key是candidate key的一种，然后剩下的在code里面是uniquekey
forerign key是来自其他

Reasoning About FDs (Cont'd.)

(Recall: "two matching X's always have the same Y")

- A few additional rules (which follow from AA):
  - *Union*: If X → Y and X → Z, then X → YZ 合起来
  - *Decomposition*: If X → YZ, then X → Y and X → Z 分开
- Example: Contracts(*cid,sid,pjid,did,pid,qty,value*), and:
  - The contract id is the key: **C** → CSJDPQV
  - A pro***j***ect purchases each ***p***art using single ***c***ontract: JP → C
  - A ***d***ept purchases at most one ***p***art from a ***s***upplier: SD → P
- JP → C, C → CSJDPQV imply JP → CSJDPQV 用了transitive
- SD → P implies SDJ → JP *(New candidate keys…!)* 用了Augmentation
- SDJ → JP, JP → CSJDPQV imply SDJ → CSJDPQV 用了Transitivity

Reasoning About FDs (Examples)

Let's consider R(ABCDE), F = {A → B, B → C, C D → E}

- Let's work our way towards inferring F+ …

(a) A→B (b) B→C (c) CD→E	(*given*)
(d) A→C	(a, b, and transitivity)
(e) BD→CD	(b and augmentation)
(f) BD→E	(e, c and transitivity)
(g) AD→CD	(d and augmentation)
(h) **AD→**E	(g, c and transitivity)
(i) **AD→**C (j) **AD→**D	(g and decomposition)
(k) AD→BD	(a and augmentation)
(l) **AD→**B	(k and decomposition)
(m) **AD→**A	(a and reflexivity)
(n) AD→ABCDE	(h, i, j, l, m, and union) ….

Candidate key!

Note: If some attribute *X* is not on the RHS of *any* initial FD, then *X* must be *part of the key*!

如果这样算出来了candidate key，但是还没有x，那么x就是其中一个key