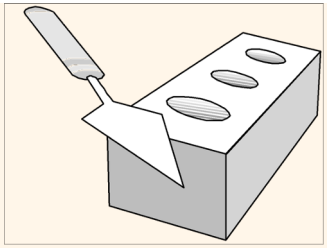
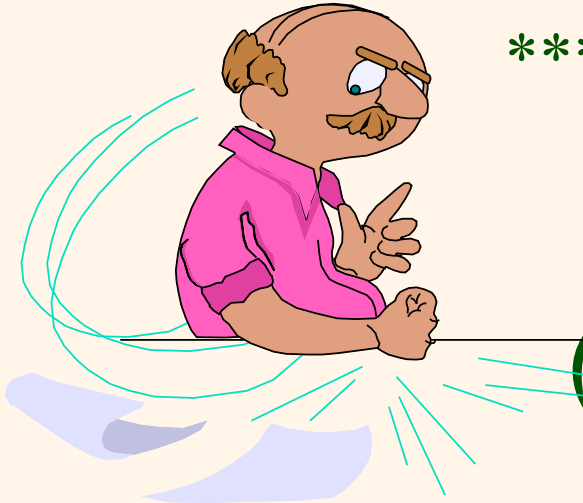


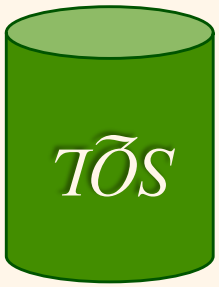
Introduction to Data Management



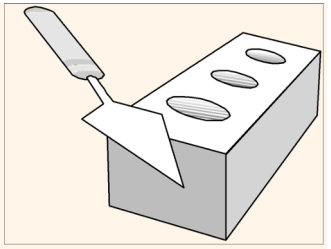
**** The "Flipped" Edition ****

Lecture #6
(Relational DB Design)

Instructor: Mike Carey
mjcarey@ics.uci.edu

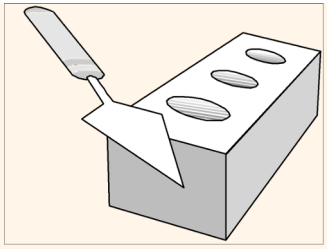


Today's Notices



- Never stop watching the wiki page:
  - http://www.ics.uci.edu/~cs122a/
- Or following our Piazza Q&A:
  - piazza.com/uci/fall2021/cs122aeecs116
- HW #1 is due today (Wednesday)
  - Thus begins a 24-hour/10-point grace period...
  - Solution will be released at 6PM Thursday (!)
- HW #2 will become available today
  - Its starting point will be HW #1's *solution*
  - Start now (get PostgreSQL & do some thinking)

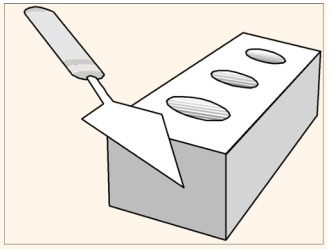
You Are Here

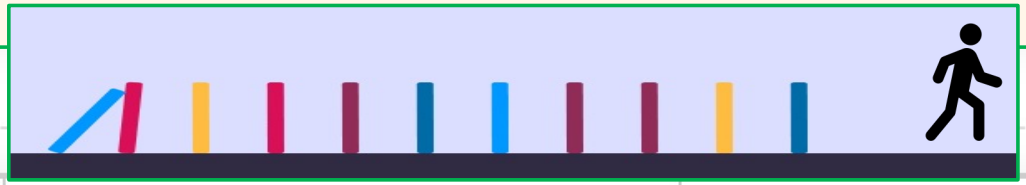


Syllabus

Topic	Reading (Required!)
Databases and DB Systems	Ch. 1
Entity-Relationship (E-R) Data Model	Ch. 6.1-6.5, 6.8-6.9
Relational Data Model	Ch. 2.1-2.4, 3.1-3.2
E-R to Relational Translation	Ch. 6.6-6.7
Relational Design Theory	Ch. 7.1-7.4.1
Midterm Exam 1	***Fri, Oct 22*** (during lecture time)
Relational Algebra	Ch. 2.5-2.7
Relational Calculus	Wikipedia: Tuple relational calculus
SQL Basics (SPJ and Nested Queries)	Ch. 3.3-3.5
SQL Analytics: Aggregation, Nulls, and Outer Joins	Ch. 3.6-3.9, 4.1
Advanced SQL: Constraints, Triggers, Views, and Security	Ch. 4.2, 4.4-4.5, 4.7
Midterm Exam 2	***Mon, Nov 15*** (during lecture time)
Storage	Ch. 12.1-12.4, 12.6-12.7
Indexing	Ch. 14.1-14.4, 14.5
Physical DB Design	Ch. 14.6-14.7, 15.1-15.3, 15.5.3
Semistructured Data Management (*a.k.a.* NoSQL)	Ch. 8.1, AsterixDB SQL++ Primer, Couchbase SQL++ Book
Data Science 1: Advanced SQL Analytics	Ch. 5.5, 11.3
Data Science 2: Notebooks, Dataframes, and Python/Pandas	Lecture notes and Jupyter notebook
Basics of Transactions	Ch. 4.3, Ch. 17
Endterm Exam	***Fri, Dec 3*** (during lecture time)

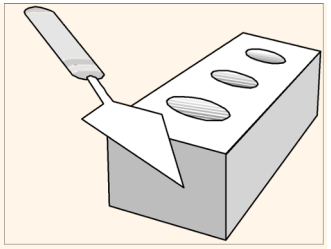
And Also Here





Homework Assignments

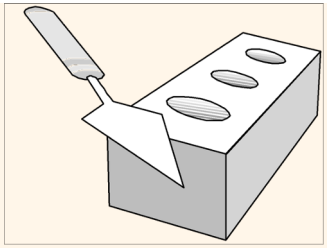
Due Date	Topic	HW Assignment
Wed, Oct 6 (6:00 PM Pacific)	E-R Modeling	HW1 Details ⇨ Template
Wed, Oct 13 (6:00 PM Pacific)	E-R to Relational Translation	
Wed, Oct 20 (6:00 PM Pacific)	Principled Relational DB Design	
Fri, Oct 29 (6:00 PM Pacific)	Relational Algebra	
Fri, Nov 5 (6:00 PM Pacific)	SQL	
Fri, Nov 12 (6:00 PM Pacific)	More SQL	
Mon, Nov 22 (6:00 PM Pacific)	Physical DB Design	
Wed, Dec 1 (6:00 PM Pacific)	NoSQL and Analytics	



Relational Database Design

- Two aspects to the RDB design problem:
  - *Logical* schema design: We just saw one approach, namely, doing E-R modeling followed by an E-R to relational schema translation step
  - *Physical* schema design: Later, once we learn about indexes, how and when should we utilize them?
- We will look at *both* RDB problem aspects this term, starting with logical schema design
  - Our power tools will be functional dependencies (FDs) and normal forms
  - Note: FDs also play an important role in other contexts as well, e.g., SQL query optimization

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

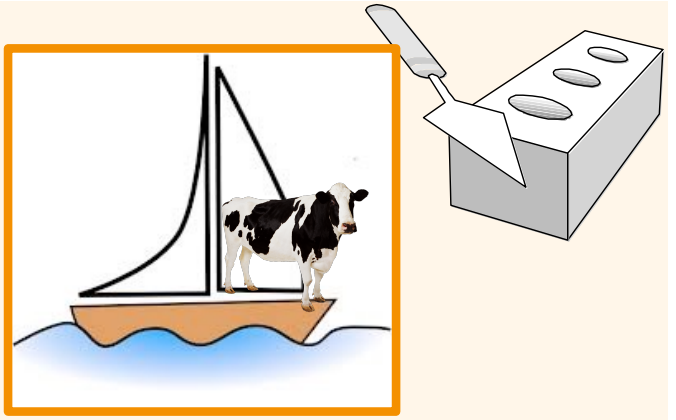
Proposed schema design #1:

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
…	…	…	…	…	…	…	…

Q: Do you think this is a "good" design? (Why or why not?)

Q: And – How to add a new Sailor, or a new Boat…?

Ex: *Wisconsin Sailing Club*



Proposed schema design #1:

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
…	…	…	…	…	…	…	…

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

Ex: *Wisconsin Sailing Club*

Proposed schema design #2:

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
…	…	…	…

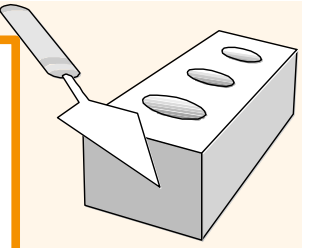
sid	bid	date
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
…	…	…

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

Q: How about *this* design?

- Is design #2 "better" than #1…? (Explain!)
- Is it a "best" design?
- How might we go from design #1 to this design?

Ex: *Wisconsin Sailing Club*



Proposed schema design #2:

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
...	...	...	...

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

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

Ex: *Wisconsin Sailing Club*

Proposed schema design #3:

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
...	...	...	...

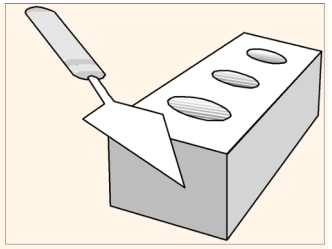
sid	bid	date
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
...	...	...

bid	bname
101	Interlake
102	Interlake
103	Clipper
104	Marine

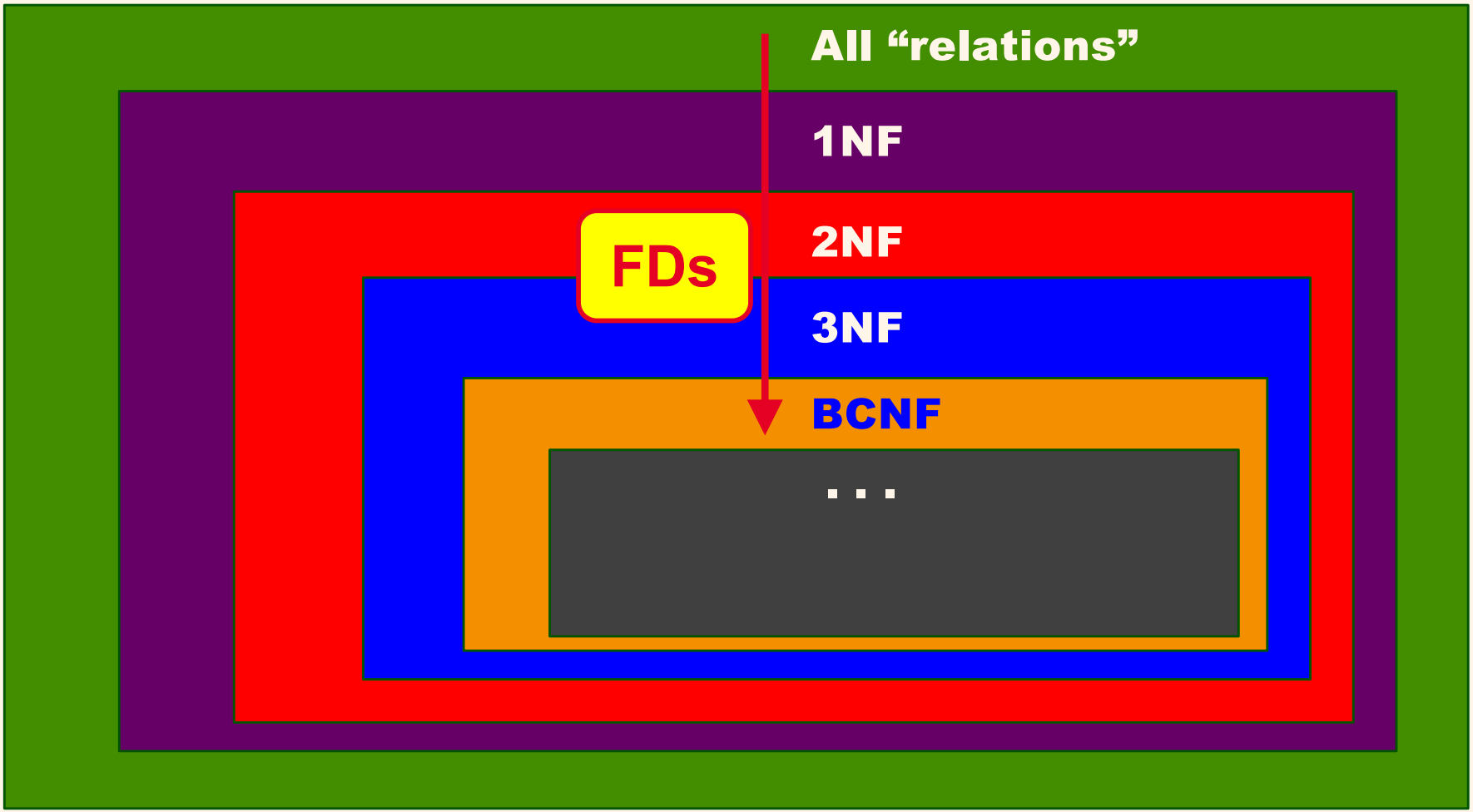
bid	color
101	blue
102	red
103	green
104	red

Q: What about ***this*** design?

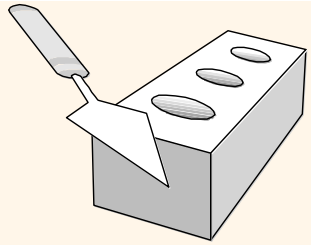
- Is design #3 "better" or "worse" than #2...?



Normal Forms (Preview)



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

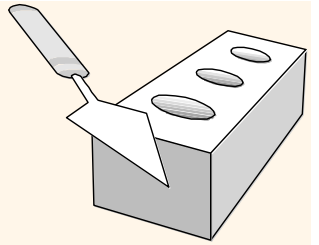


- *Redundancy* is at the root of several problems associated with relational schemas:
  - Redundant storage (space)
  - Insert/delete/update anomalies

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

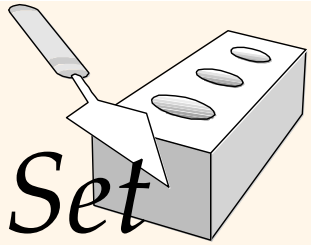
- *Functional dependencies* can help in identifying problem schemas and suggesting refinements.
- Main refinement technique: *decomposition*, e.g., replace R(ABCD) with R1(AB) + R2(BCD)
- Decomposition should be used judiciously:
  - Is there reason to decompose a relation?
  - Does the decomposition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

Functional Dependencies (FDs)



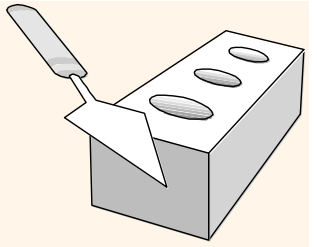
- A functional dependency X → Y holds over relation R if, for every allowable instance *r* of R:
  - For *t1* ∈ *r* and *t2* in *r*, *t1*.X = *t2*.X implies *t1*.Y = *t2*.Y
  - I.e., *given two tuples in r, if the X values agree, then their Y values must also agree*. (X and Y can be *sets* of attributes.)
- 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 *r1* violates some FD *f*, but we *cannot* tell if *f* holds over R!
- Saying K is a candidate key for R means 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 "*superkey*").

Example: Constraints on an Entity Set



- Suppose you're given a relation called HourlyEmps:
  - HourlyEmps (*ssn*, *name*, *lot*, *rating*, *hrly_wages*, *hrs_worked*)
- *Notation*: 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 vs. SNLRWH).
- Suppose we also have some FDs on HourlyEmps:
  - *ssn* is the key: S → SNLRWH
  - *rating* determines *hrly_wages*: R → W

Example (Cont'd.)



Wages

R	W
8	10
5	7

+

HourlyEmps2

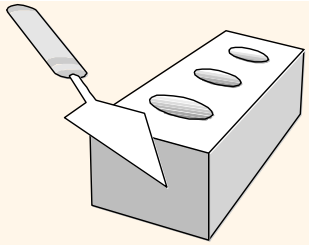
S	N	L	R	H
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40

Normalize

- 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?

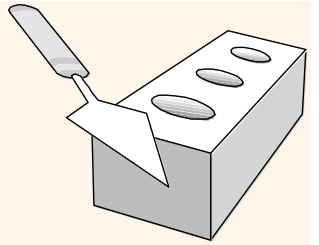
S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40



What FD's Are and Aren't

- X → Y: Read this as "X determines Y", or as "Y is functionally dependent on X"
- FD's ***are*** a form of consistency constraint
  - *Ex:* email → name implies that whenever a given email address appears in the DB, it will be (or should be!) associated with the same name if they appear together – like mjcarey@ics.uci.edu and "Michael J. Carey"
- X → Y ***does not*** mean we have a function f(X) that we can use to compute Y from X
  - The "function" is conceptual (i.e., it's "as if...")

Again: Consider *rating* → *hrly_wages*



To Be Continued...