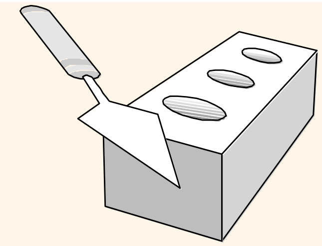
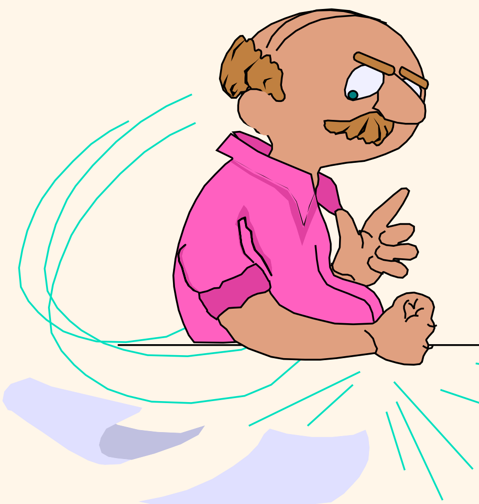


Introduction to Data Management

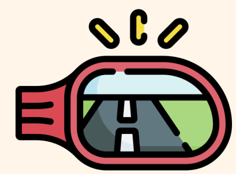
*** *The "Flipped" Edition* ***

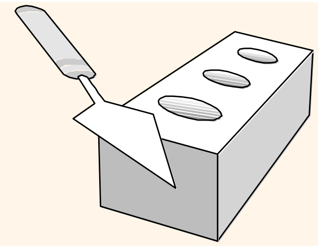


Lecture #7
(Relational Design Theory)

Instructor: Mike Carey
mjcarey@ics.uci.edu

Today's Notices

- Keep one eye on the wiki page…
  - http://www.ics.uci.edu/~cs122a/
- … and the other eye on Piazza Q&A!
  - piazza.com/uci/fall2021/cs122aeecs116
- HW #1 is now in the rearview mirror 
  - Thursday at 6PM is/was the drop-dead deadline
  - Ask questions in "lecture"/discussions/Piazza
- HW #2 is your next destination 
  - Its starting point is ***HW #1's solution (!)***
  - We never want you to be bored in this class…



Quick Roadmap Check...

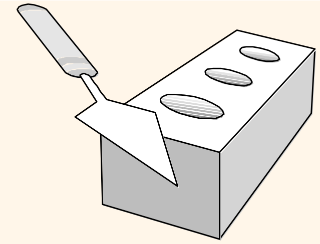
Topic Coverage and Exam Schedule

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)

Midterm Exam 1

Time: Fri, Oct 22, Lecture Time
Place: SSLH 100



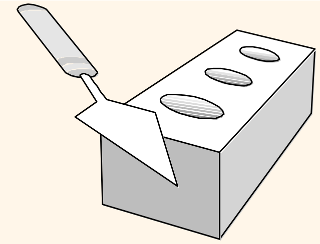
Reasoning About FDs

- Given some FDs, we can usually infer additional FDs:
  - $ssn \rightarrow did,\ did \rightarrow lot$ implies $ssn \rightarrow lot$
  - (Translation: *M*atching *ssn*s imply matching *lot*s.)
- An FD *f* is ***implied by*** a set of FDs *F* if *f* holds whenever all FDs in *F* hold.
  - F^+ = *closure of F* is the set of *all* FDs that are implied by *F*.
- **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 $X \subseteq Y$ then $Y \rightarrow X$:
  - **zip** $\subseteq$ (**zip**, name), so (**zip**, name) $\rightarrow$ **zip.**
- *Augmentation*: If $X \rightarrow Y$ then $XZ \rightarrow YZ$ for any Z:
  - zip $\rightarrow$ state, so (zip, **title**) $\rightarrow$ (state, **title**).
- *Transitivity*: If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$:
  - **pno** $\rightarrow$ zip and zip $\rightarrow$ **state**, so **pno** $\rightarrow$ **state.**



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 $\boxed{X \to Y}$ and $X \to Z$, then $X \to YZ$
  - *Decomposition:* If $X \to YZ$, then $X \to Y$ and $X \to Z$
- Example: Contracts(*cid,sid,pjid,did,pid,qty,value*), and:
  - The ***c***ontract id is the key: **C** $\to$ CSJDPQV
  - A pro***j***ect purchases each ***p***art using single ***c***ontract: JP $\to$ C
  - A ***d***ept purchases at most one ***p***art from a ***s***upplier: SD $\to$ P
- JP $\to$ C, C $\to$ CSJDPQV imply JP $\to$ CSJDPQV
- SD $\to$ P implies SDJ $\to$ JP

(New candidate keys…!)

- SDJ $\to$ JP, JP $\to$ CSJDPQV imply SDJ $\to$ CSJDPQV

Reasoning About FDs (Examples)

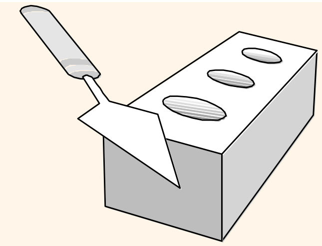
Let's consider R(ABCDE), F = {A → B, B → C, C D → E}

- 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*!



Reasoning About FDs (Cont'd.)

- Computing the closure of a set of FDs can be very expensive. (Closure size is exponential in # of attrs!)
- Typically, we just want to check if a ***specific*** FD *X* → *Y* is ***in*** the closure of a set of FDs *F*. An efficient check:
  - **First**: Compute *attribute closure* of *X* (denoted **X+**) w.r.t. *F*:
    - Set of all attributes *A* such that *X* → *A* is in *F*+ (i.e., all *F*+ attributes)
    - There is a *linear time algorithm* to compute this (look here): Start with *X* and keep adding attributes that can (now) be inferred via the FDs
  - **Then**: Check to see if Y is in X+
- Does F = {A → B, B → C, C D → E } imply A → E?
  - I.e.: Is A → E in the closure **F+**? Equivalently: Is E in **A+**?

FDs & Redundancy

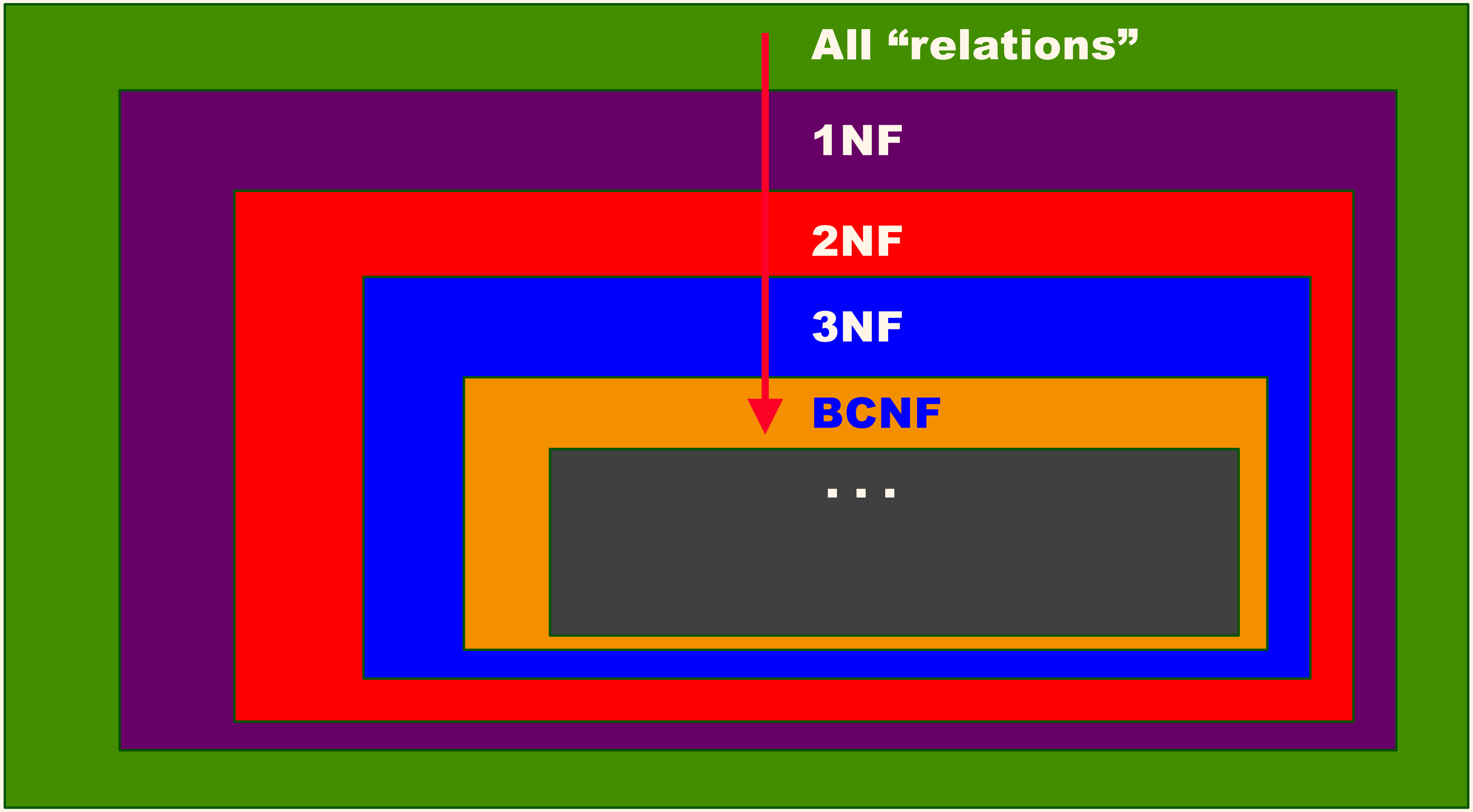
- Role of FDs in detecting redundancy in a schema:
  - Consider a relation R with three attributes, say R(ABC).
    - If **no** (non-trivial) FDs hold: There is *no redundancy* here then. (Think about this … in fact, think ***hard***…!)
      - *Ex:* Prescriptions(doc_name, patient_name, drug_name)
    - Given A → B: Several tuples could have the same A value – and if so, then they'll all have the same B value as well! Thus, if A is repeated for some reason, it will always have the same B "tagging along for the ride".
      - **Ex:** Employee(emp_name, dept_no, mgr_name)

        (Redundancy here if dept_no → mgr_name!)

Normal Forms

- Returning to the issue of schema refinement, the first question to ask is whether any refinement is needed!
- We will define various *normal forms* (BCNF, 3NF etc.) based on the nature of FDs that hold.
- Depending upon the normal form a relation is in, it has a different level of redundancy.
  - E.g., a BCNF relation has NO redundancy (as you'll learn).
- Checking which normal form a given relation is in will help us decide if we need to decompose (fix) it.
  - E.g., there's no need to decompose a BCNF relation!

Normal Forms

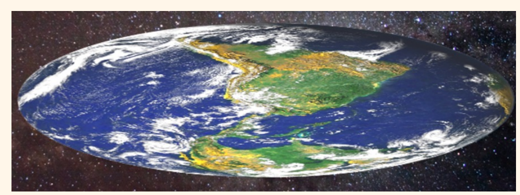


First Normal Form (1NF)

- Rel'n R is in 1NF if all of its attributes are atomic.
  - No set-valued attributes! (1NF = "flat" ☺)
  - Usually goes *w/o* saying for relational model (but not for *NoSQL* systems, as we'll see at the end of the quarter ☺).
  - *Ex:*

bname	color
Interlake	blue, red
Clipper	green
Marine	red

bname	color
Interlake	blue
Interlake	red
Clipper	green
Marine	red



(1NF is different than the other normal forms.)

Some Terms and Definitions (Review)

- If X is part of a (candidate) key, we will say that X is a *prime attribute*.
- If X (an attribute set) contains a candidate key, we will say that X is a *superkey*.
- X → Y can be pronounced as "X *determines* Y", or "Y is *functionally dependent* on X".
- Some types of dependencies (*on a key*):
  - *Trivial*: **XY → X**
  - *Partial*: **X**Y is a key, **X → Z** (note that Y is absent)
  - *Transitive*: X → Y, Y → Z, Y is non-prime, **X → Z**

To Be Continued….