Announcements

- **Homework stuff**
  - HW #2 is due this Friday before class
  - HW #3 will be due the following **Wednesday**
- **Exam stuff (time flies!)**
  - Midterm #1 is a week from Friday (**in class**)
  - We’ll use assigned seating (more info next week), so you’ll want to try and show up early to get settled in
  - An 8.5”x11” 2-sided cheat sheet will be permitted
- **Today’s plan:**
  - Relational DB design theory (**II**)
  - **Disclaimer:** Still not the most exciting CS122A topic… 😊
Reasoning About FDs (Examples)

Let’s consider \( R(ABCDE) \), \( F = \{ A \to B, B \to C, CD \to E \} \):

- Let’s work our way towards inferring \( F^+ \) ...
  
  (a) \( A \to B \)
  
  (b) \( B \to C \)
  
  (c) \( CD \to E \)  \( \text{(given)} \)
  
  (d) \( A \to C \)  \( \text{(a, b, and transitivity)} \)
  
  (e) \( BD \to CD \)  \( \text{(b and augmentation)} \)
  
  (f) \( BD \to E \)  \( \text{(e, c and transitivity)} \)
  
  (g) \( AD \to CD \)  \( \text{(d and augmentation)} \)
  
  (h) \( AD \to E \)  \( \text{(g, c and transitivity)} \)
  
  (i) \( AD \to C \)  \( \text{(g and decomposition)} \)
  
  (j) \( AD \to D \)  \( \text{(g and decomposition)} \)
  
  (k) \( AD \to BD \)  \( \text{(a and augmentation)} \)
  
  (l) \( AD \to B \)  \( \text{(k and decomposition)} \)
  
  (m) \( AD \to A \)  \( \text{(a and reflexivity)} \)
  
  (n) \( AD \to ABCDE \)  \( \text{(h, i, j, l, m, and union)} \)  \( \text{Candidate key!} \)

\[ \text{Note: If some attribute X is not on the RHS of any initial FD, then X must be part of the key!} \]

Reasoning About FDs (Cont’d.)

- Computing the closure of a set of FDs can be expensive. (Size of closure is exponential in # attrs!)
- Typically, we just want to check if a given FD \( X \to Y \) 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 \( X \to A \) is in \( F^+ \) (i.e., all \( F^+ \) attributes)
    
    - There is a linear time algorithm to compute this: start with \( X \) and keep adding attributes that can be inferred via the FDs.
  
  - Then check to see if \( Y \) is in \( X^+ \)

- \( F = \{ A \to B, B \to C, CD \to E \} \) imply \( A \to E \)?
  
  - I.e.: Is \( A \to E \) 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 3 attributes, ABC.
    - **If no (non-trivial) FDs hold:** There is no redundancy here then. (Think about this – in fact, think hard...!)
    - **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”.)

**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 different level of redundancy
  - E.g., a BCNF relation has NO redundancy – clearer soon!
- Checking for which normal form a relation is in will help us decide whether to decompose the relation
  - E.g., no point in decomposing a BCNF relation!
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**: XY is a key, X → Z
  - **Transitive**: X → Y, Y → Z, Y is non-prime, X → Z
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:

<table>
<thead>
<tr>
<th>bname</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interlake</td>
<td>blue, red</td>
</tr>
<tr>
<td>Clipper</td>
<td>green</td>
</tr>
<tr>
<td>Marine</td>
<td>red</td>
</tr>
</tbody>
</table>

Second Normal Form (2NF)

- Rel’n R is in 2NF if it is in 1NF and no non-prime attribute is partially dependent on a candidate key of R.
- Ex: Supplies(sno, sname, saddr, pno, pname, pcolor)
  where: sno → sname, sno → saddr, pno → pname, pno → pcolor

Q1: What are the candidate keys for Supplies?
Q2: What are the prime attributes for Supplies?
Q3: Why is Supplies not in 2NF?
Q4: What’s the fix?

Supplier(sno, sname, saddr)
Part(pno, pname, pcolor)
Supply(sno, pno)

A1: (sno, pno)
A2: sno, pno
A3: Each of its four FDs violates 2NF!

Must not forget this!
(Else “lossy join”!!)