Science is the knowledge of consequences, and dependence of one fact upon another.

Thomas Hobbes
(1588-1679)

**Functional Dependencies**

R&G Chapter 19

**Review: Database Design**
- Requirements Analysis
  - user needs; what must database do?
- Conceptual Design
  - high level descr (often done w/ER model)
- Logical Design
  - translate ER into DBMS data model
- Schema Refinement
  - consistency, normalization
- Physical Design - indexes, disk layout
- Security Design - who accesses what

**The Evils of Redundancy**

- **Redundancy** is at the root of several problems associated with relational schemas:
  - redundant storage, insert/delete/update anomalies
- Integrity constraints, in particular **functional dependencies**, can be used to identify schemas with such problems and to suggest refinements.
- Main refinement technique: **decomposition**
  - replacing ABCD with, say, AB and BCD, or ACD and ABD.
- Decomposition should be used judiciously:
  - Is there reason to decompose a relation?
  - What problems (if any) does the decomposition cause?

**Functional Dependencies (FDs)**

- A functional dependency \( X \rightarrow Y \) holds over relation schema \( R \) if, for every allowable instance \( r \) of \( R \):
  \[
  t_1 \in r, \quad t_2 \in r, \quad \mu_X(t_1) = \mu_X(t_2) \implies \mu_Y(t_1) = \mu_Y(t_2)
  \]
  (where \( t_1 \) and \( t_2 \) are tuples; \( X \) and \( Y \) are sets of attributes)

- In other words: \( X \rightarrow Y \) means
  Given any two tuples in \( r \), if the \( X \) values are the same, then the \( Y \) values must also be the same.
  (but not vice versa)

- **Can read** \( \rightarrow \) **as** “determines”

**FD’s Continued**

- An FD is a statement about **all** allowable relations.
  - Must be identified based on semantics of application.
  - Given some instance \( r \) of \( R \), we can check if \( r \) violates some FD \( f \), but we cannot determine if \( f \) holds over \( R \).

- **Question: How related to keys?**
  - if \( "K \rightarrow all attributes of R" \) then \( K \) is a **superkey** for \( R \)
    (does not require \( K \) to be **minimal**.)

- **FDs are a generalization of keys.**

**Example: Constraints on Entity Set**

- Consider relation obtained from Hourly_Emps: \( \text{Hourly}_\text{Emps} \) (\( \text{ssn}, \text{name}, \text{lot}, \text{rating}, \text{wage}\_\text{per}\_\text{hr}, \text{hrs}_\text{per}_W \))

  - We sometimes denote a relation schema by listing the attributes: e.g., \( \text{SNLRWH} \)
  - This is really the set of attributes \( \{S,N,L,R,W,H\} \).

  - Sometimes, we refer to the set of all attributes of a relation by using the relation name. e.g., “Hourly_Emps” for SNLRWH

- What are some FDs on Hourly_Emps?

  - \( \text{ssn} \text{ is the key: } S \rightarrow \text{SNLRWH} \)
  - \( \text{rating determines } \text{wage}\_\text{per}\_\text{hr: } R \rightarrow W \)
  - \( \text{lot determines } \text{lot: } L \rightarrow L \) (“trivial” dependency)
Problems Due to $R \rightarrow W$

- **Update anomaly:** Can we modify W in only the 1st tuple of SNLRWH?
- **Insertion anomaly:** What if we want to insert an employee and don’t know the hourly wage for his or her rating? (or we get it wrong?)
- **Deletion anomaly:** If we delete all employees with rating 5, we lose the information about the wage for rating 5!

<table>
<thead>
<tr>
<th>S</th>
<th>N</th>
<th>L</th>
<th>R</th>
<th>W H</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldu</td>
<td>35</td>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>612-67-4134</td>
<td>Madayan</td>
<td>35</td>
<td>8</td>
<td>40</td>
</tr>
</tbody>
</table>

Detecting Reduncancy

<table>
<thead>
<tr>
<th>S</th>
<th>N</th>
<th>L</th>
<th>R</th>
<th>W H</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldu</td>
<td>35</td>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>612-67-4134</td>
<td>Madayan</td>
<td>35</td>
<td>8</td>
<td>40</td>
</tr>
</tbody>
</table>

Q: Why was $R \rightarrow W$ problematic, but $S \rightarrow W$ not?

Decomposing a Relation

- Redundancy can be removed by "chopping" the relation into pieces.

- FD’s are used to drive this process.

  $R \rightarrow W$ is causing the problems, so decompose SNLRWH into what relations?

<table>
<thead>
<tr>
<th>S</th>
<th>N</th>
<th>L</th>
<th>R</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldu</td>
<td>35</td>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>612-67-4134</td>
<td>Madayan</td>
<td>35</td>
<td>8</td>
<td>40</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>N</th>
<th>L</th>
<th>R</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldu</td>
<td>35</td>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>612-67-4134</td>
<td>Madayan</td>
<td>35</td>
<td>8</td>
<td>40</td>
</tr>
</tbody>
</table>

Refining an ER Diagram

- 1st diagram becomes:
  - Workers(S,N,L,D,Si)
  - Departments(D,M,B)
  - Lots associated with workers.

- Suppose all workers in a dept are assigned the same lot: $D \rightarrow L$

- Redundancy; fixed by:
  - Workers2(S,N,D,Si)
  - Dept_Lots(D,L)
  - Departments(D,M,B)

- Can fine-tune this:
  - Workers2(S,N,D,Si)
  - Departments(D,M,B,L)

Reasoning About FDs

- Given some FDs, we can usually infer additional FDs:
  - $title \rightarrow studio, star$ implies $title \rightarrow studio$ and $title \rightarrow star$
  - $title \rightarrow studio$ and $title \rightarrow star$ implies $title \rightarrow studio, star$
  - $title \rightarrow studio, studio \rightarrow star$ implies $title \rightarrow star$

But,

- $title, star \rightarrow studio$ does NOT necessarily imply that $title \rightarrow studio$ or that $star \rightarrow studio$

- 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 $F$. (includes "trivial dependencies")

Rules of Inference

- **Armstrong’s Axioms** ($X, Y, Z$ are sets of attributes):
  - Reflexivity: If $X \rightarrow Y$, then $X \rightarrow Y$
  - 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! – i.e., using AA you can compute all the FDs in $F^+$ and only these FDs.

- Some additional rules (that follow from AA):
  - Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
  - Decomposition: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
Example
• Contracts(cid, sid, did, pid, qty, value), and:
  – C is the key: C \rightarrow CSJPQV
  – Proj purchases each part using single contract: JP \rightarrow C
  – Dept purchases at most 1 part from a supplier: SD \rightarrow P
• Problem: Prove that SDJ is a key for Contracts
  • JP \rightarrow C, C \rightarrow CSJPQV imply JP \rightarrow CSJPQV
    (by transitivity) (shows that JP is a key)
  • SD \rightarrow P implies SDJ \rightarrow JP
    (by augmentation)
  • SDJ \rightarrow JP, JP \rightarrow CSJPQV imply SDJ \rightarrow CSJPQV
    (by transitivity) thus SDJ is a key.
Q: can you now infer that SD \rightarrow CSDPQV (i.e., drop J on both sides)?
No! FD inference is not like arithmetic multiplication.

Attribute Closure
• 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 \rightarrow Y is in the closure of a set of FDs F. An efficient check:
  – Compute attribute closure of X (denoted X+) wrt F.
    \[ X^+ = \text{Set of all attributes } A \text{ such that } X \rightarrow A \text{ is in } F^+ \]
  – X^+ := X
  – Repeat until no change: if there is an fd U \rightarrow V in F such that U is in X^+, then add V to X^+
  – Check if Y is in X^+
• Approach can also be used to find the keys of a relation.
  • If all attributes of R are in the closure of X then X is a superkey for R.
  • Q: How to check if X is a “candidate key”?

Attribute Closure (example)
• R = \{A, B, C, D, E\}
• F = \{B \rightarrow CD, D \rightarrow E, B \rightarrow A, E \rightarrow C, AD \rightarrow B\}
• Is B \rightarrow E in F^+?
  B^+ = B
  B^+ = BCD
  B^+ = BCDA
  B^+ = BCDAE ... Yes! and B is a key for R too!
• Is D a key for R?
  D^+ = D
  D^+ = DE
  D^+ = DEC
  ... Nope!
• Is AD a key for R?
  AD^+ = AD
  AD^+ = ABD and B is a key, so Yes!
• Is AD a candidate key for R?
  A^+ = A, D^+ = DEC
  ... A,D not keys, so Yes!
• Is ADE a candidate key for R?
  ... Not! AD is a key, so ADE is a superkey, but not a cand. key

Next Class...
• Normal forms and normalization
• Table decompositions