Outline

- Functional dependencies (FD)
- Properties of FD
- Inferring FD
- Normalization
  - 3NF
  - BCNF
- Multivalued dependencies
  - 4NF

Functional Dependencies

- $X \rightarrow A$
  - assertion about a relation $R$ that whenever two tuples agree on all the attributes of $X$, then they must also agree on attribute $A$.
  - Important as a constraint on the data that may appear within a relation.
  - Schema-level control of data.
  - Mathematical tool for explaining the process of "normalization"—vital for redesigning database schemas when original design has certain flaws.

FD Conventions

- $X$, etc., represent sets of attributes; $A$ etc., represent single attributes.
- No set formers ({}), e.g., $\{A, B, C\}$.

Example

<table>
<thead>
<tr>
<th>Drinks(name, addr, beersLiked, manf, favoriteBeer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
</tr>
<tr>
<td>Mike</td>
</tr>
<tr>
<td>Mike</td>
</tr>
<tr>
<td>Anna</td>
</tr>
</tbody>
</table>

- Reasonable FD's to assert:
  1. ...
  2. ...
  3. ...

- Note: FD's can give more detail than just assertion of a key.

Properties of FD's

- Key (in general) functionally determines all attributes. In our example: $name \text{ beersLiked} \rightarrow addr \text{ favoriteBeer} \text{ beerManf}$
- Shorthand: combine FD's with common left side by concatenating their right sides.
- When FD's are not of the form Key $\rightarrow$ other attribute(s), then there is typically an attempt to "cram" too much into one relation.
Properties of FD’s

- Sometimes, several attributes jointly determine another attribute, although neither does by itself.
- Example: \( \text{beer bar} \rightarrow \text{price} \)

Formal Notion of Key

- \( K \) is a key for relation \( R \) if:
  1. \( K \rightarrow \) all attributes of \( R \).
  2. For no proper subset of \( K \) is (1) true.
- If \( K \) at least satisfies (1), then \( K \) is a superkey.

Example

\( \text{Drinkers(name, addr, beersLiked, manf, favoriteBeer)} \)

- \( \{\text{name, beersLiked}\} \) FD’s all attributes, as seen.
  - Shows \( \{\text{name, beersLiked}\} \) is a superkey.
- \( \text{name} \rightarrow \text{beersLiked} \) is false, so name not a superkey.
- \( \text{beersLiked} \rightarrow \text{name} \) also false, so beersLiked not a superkey.
- Thus, \( \{\text{name, beersLiked}\} \) is a key.
- No other keys in this example.
  - Neither name nor beersLiked is on the right of any observed FD, so they must be part of any superkey.

Who Determines Keys/FD’s?

- We could define a relation schema by simply giving a single key \( K \).
  - Then the only FD’s asserted are that \( K \rightarrow A \) for every attribute \( A \).
  - No surprise: \( K \) is then the only key for those FD’s, according to the formal definition of “key.”
- Or, we could assert some FD’s and deduce one or more keys by the formal definition.
  - E/R diagram implies FD’s by key declarations and many-one relationship declarations.
  - Rule of thumb: FD’s either come from keyness, many-1 relationship, or from physics.
  - E.g., “no two courses can meet in the same room at the same time” yields \( \text{room time} \rightarrow \text{course} \).

Inferring FD’s

- When we talk about improving relational designs, we often need to ask “does this FD hold in this relation?”
- Given FD’s \( X_1 \rightarrow A_1, \cdots, \cdots, X_n \rightarrow A_n \), does FD \( Y \rightarrow B \) necessarily hold in the same relation?
- Start by assuming two tuples agree in \( Y \). Use given FD’s to infer other attributes on which they must agree. If \( B \) is among them, then yes, else no.

Closure of Attributes

- Given a relation \( R \) with attributes \( X \) and a subset of the attributes \( Y \).
- Find all \( A \)’s such that \( Y \rightarrow A \).
- Define \( Y^+ = \text{closure of } Y = \text{set of attributes functionally determined by } Y \) (all the A’s)
**Closure Algorithm**

- **Basis:** $Y^+ := Y$.
- **Induction:** If $X \subseteq Y^+$ and $X \rightarrow A$ is a given FD, then add $A$ to $Y^+$.
- **End when $Y^+$ cannot be changed.**

**Example**

- Relation $R(A,B,C,D)$.
- FD's: $A \rightarrow B$, $BC \rightarrow D$.
- $A^+ = AB$.
- $C^+ = C$.
- $(AC)^+ = ABCD$.

**Given Versus Implied FD's**

- Typically, we state a few FD's that are known to hold for a relation $R$.
- Other FD's may follow logically from the given FD's; these are **implied** FD's.
- We are free to choose any **basis** for the FD's of $R$ – a set of FD's that imply all the FD's that hold for $R$.

**Finding All Implied FD's**

- Motivation: Suppose we have a relation $ABCD$ with some FD's $F$. If we decide to decompose $ABCD$ into $ABC$ and $AD$, what are the FD's for $ABC$, $AD$?
- **Example:** $F = AB \rightarrow C$, $C \rightarrow D$, $D \rightarrow A$.
  - It looks like just $AB \rightarrow C$ holds in $ABC$, but in fact $C \rightarrow A$ follows from $F$ and applies to relation $ABC$.
- Problem is exponential in worst case.

**Algorithm**

- For each set of attributes $X$ compute $X^+$.
- Add $X \rightarrow A$ for each $A$ in $X^+\setminus X$.
- Ignore or drop some "obvious" dependencies that follow from others:
  1. **Trivial FD's:** right side is a subset of left side.
     - Consequence: no point in computing $\overline{X}^+$ or closure of full set of attributes.
  2. Drop $XY \rightarrow A$ if $X \rightarrow A$ holds.
     - Consequence: if $X^+$ is all attributes, then there is no point in computing closure of supersets of $X$.
  3. Ignore FD's whose right sides are not single attributes.
- Notice that after we project the discovered FD's onto some relation, the FD's eliminated by rules 1, 2, and 3 can be inferred in the projected relation.

**Example**

- $F = AB \rightarrow C$, $C \rightarrow D$, $D \rightarrow A$. What FD's follow?
  - $A^+ = A$; $B^+ = B$ (nothing).
  - $C = ACD$ (add $C \rightarrow A$).
  - $D^+ = AD$ (nothing new).
  - ...
Normalization

- Improve the schema by decomposing relations and removing anomalies.
- Boyce-Codd Normal Form (BCNF): all FD’s follow from the fact key → everything.
- Formally, R is in BCNF if every nontrivial FD for R, say X → A, has X a superkey.
  - "Nontrivial" = right-side attribute not in left side.

BCNF properties

1. Guarantees no redundancy due to FD’s.
2. Guarantees no update anomalies = one occurrence of a fact is updated, not all.
3. Guarantees no deletion anomalies = valid fact is lost when tuple is deleted.

Example (1/2)

**Drinkers**

<table>
<thead>
<tr>
<th>name</th>
<th>addr</th>
<th>beersLiked</th>
<th>manf</th>
<th>favoriteBeer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>111 E Ohio</td>
<td>Bud</td>
<td>A.B.</td>
<td>Blonde Ale</td>
</tr>
<tr>
<td>Mike</td>
<td>???</td>
<td>Blonde Ale</td>
<td>G.I.</td>
<td>???</td>
</tr>
<tr>
<td>Anna</td>
<td>123 W Grand</td>
<td>Bud</td>
<td>???</td>
<td>BudLite</td>
</tr>
</tbody>
</table>

- FD’s:
  1. name → addr
  2. name → favoriteBeer
  3. beersLiked → manf
- ???’s are redundant, since we can figure them out from the FD’s.
- Update anomalies: If Mike moves, we need to change addr in each of his tuples?
- Deletion anomalies: If nobody likes Bud, we lose track of Bud’s manufacturer.

Example (2/2)

Each of the given FD’s is a BCNF violation:
- Key = \{name, beersLiked\}
  - Each of the given FD’s has a left side a proper subset of the key.

Another Example

**Beers**

<table>
<thead>
<tr>
<th>name</th>
<th>manf</th>
<th>manfAddr</th>
</tr>
</thead>
<tbody>
<tr>
<td>manf</td>
<td></td>
<td>manfAddr</td>
</tr>
<tr>
<td>name</td>
<td>manf</td>
<td>manfAddr</td>
</tr>
</tbody>
</table>

- FD’s:
  - name → manf
  - manf → manfAddr
- Only key is name.
- manf → manfAddr violates BCNF with a left side unrelated to any key.

Decomposition into BCNF

- Setting: relation R, given FD’s F. Suppose relation R has BCNF violation X → B.
- We need only look among FD’s of F for a BCNF violation.
- If there are no violations, then the relation is in BCNF.
- Don’t we have to consider implied FD’s?
- No, because...
Proof

- Let \( Y \rightarrow A \) is a BCNF violation and follows from \( F \).
- Then the computation of \( Y^+ \) used at least one FD \( X \rightarrow B \) from \( F \).
- \( X \) must be a subset of \( Y \).
- Thus, if \( Y \) is not a superkey, \( X \) cannot be a superkey either, and \( X \rightarrow B \) is also a BCNF violation.

Decomposition Algorithm (1/2)

For every violation \( X \rightarrow B \) among given FD's:
1. Compute \( X^+ \).
   - Cannot be all attributes – why?
2. Decompose \( R \) into \( X^+ \) and \( (R-X^+) \cup X \).

Decomposition Algorithm (2/2)

3. Find the FD’s for the decomposed relations.
   - Project the FD’s from \( F = \) calculate all consequents of \( F \) that involve only attributes from \( X^+ \) or only from \( (R-X^+) \cup X \).

Example (1/3)

\( R = \) Drinkers(name, addr, beersLiked, manf, favoriteBeer)

FD's:
- name \( \rightarrow \) addr
- name \( \rightarrow \) favoriteBeer
- beersLiked \( \rightarrow \) manf

Pick BCNF violation name \( \rightarrow \) addr.
- Close the left side: name \( ^+ = \) name addr favoriteBeer.
- Decomposed relations:
  - Drinkers1(name, addr, favoriteBeer)
  - Drinkers2(name, beersLiked, manf)
- Projected FD's (skipping a lot of work):
  - For Drinkers1: name \( \rightarrow \) addr and name \( \rightarrow \) favoriteBeer.
  - For Drinkers2: beersLiked \( \rightarrow \) manf.

Example (2/3)

- BCNF violations?
  - For Drinkers1, name is key and all left sides of FD's are superkeys.
  - For Drinkers2, \( \{ \text{name, beersLiked} \} \) is the key, and beersLiked \( \rightarrow \) manf violates BCNF.

Example (3/3)

- Decompose Drinkers2
- Close beersLiked \( ^+ = \) beersLiked, manf.
- Decompose:
  - Drinkers3(beersLiked, manf)
  - Drinkers4(name, beersLiked)
- Resulting relations are all in BCNF:
  - Drinkers1(name, addr, favoriteBeer)
  - Drinkers3(beersLiked, manf)
  - Drinkers4(name, beersLiked)
Third Normal Form (3NF)

- Sometimes we have a dilemma:
  - If you decompose, you can’t check the FD’s in the decomposed relations.
  - If you don’t decompose, you violate BCNF.
- Abstractly: \( AB \rightarrow C \) and \( C \rightarrow B \).
- In book: \( \text{title} \text{ city} \rightarrow \text{theatre} \) and \( \text{theatre} \rightarrow \text{city} \).
- Another example: \( \text{street} \text{ city} \rightarrow \text{zip} \), \( \text{zip} \rightarrow \text{city} \).
- Keys: \( AB \) and \( AC \), but \( C \rightarrow B \) has a left side not a superkey.
- Suggests decomposition into \( BC \) and \( AC \).
  - But you can’t check the FD \( AB \rightarrow C \) in these relations.

Example

- What can go wrong if we decompose:
  - \( A = \text{street}, \)
  - \( B = \text{city}, \)
  - \( C = \text{zip}. \)

\[
\begin{array}{|c|c|}
\hline
\text{street} & \text{zip} \\
\hline
545 \text{ Tech Sq.} & 02138 \\
545 \text{ Tech Sq.} & 02139 \\
\hline
\text{city} & \text{zip} \\
\hline
\text{Cambridge} & 02138 \\
\text{Cambridge} & 02139 \\
\hline
\end{array}
\]

Join:

\[
\begin{array}{|c|c|c|}
\hline
\text{city} & \text{street} & \text{zip} \\
\hline
\text{Cambridge} & 545 \text{ Tech Sq.} & 02138 \\
\text{Cambridge} & 545 \text{ Tech Sq.} & 02139 \\
\hline
\end{array}
\]

Elegant Workaround

- Define the problem away.
- A relation \( R \) is in 3NF iff for every nontrivial FD \( X \rightarrow A \), either:
  1. \( X \) is a superkey, or
  2. \( A \) is prime = member of at least one key.
- Thus, the canonical problem goes away: you don’t have to decompose because all attributes are prime.

Decomposition Properties

1. We should be able to recover from the decomposed relations the data of the original.
   - Recovery involves projection and join (next time).
2. We should be able to check that the FD’s for the original relation are satisfied by checking the projections of those FD’s in the decomposed relations.

3NF vs. BCNF

- Without proof, we assert that it is always possible to decompose into BCNF and satisfy (1).
- Also without proof, we can decompose into 3NF and satisfy both (1) and (2).
- But it is not possible to decompose into BCNF and get both (1) and (2).
- Street-city-zip is an example of this point.

Multivalued Dependencies

- The multivalued dependency \( X \rightarrow Y \) holds in a relation \( R \) if whenever we have two tuples of \( R \) that agree in all the attributes of \( X \), then we can swap their \( Y \) components and get two new tuples that are also in \( R \).
Example

- Drinkers(name, addr, phones, beersLiked)
- MVD name \( \rightarrow \rightarrow \) phones.
- If Drinkers has the two tuples:

<table>
<thead>
<tr>
<th>name</th>
<th>addr</th>
<th>phones</th>
<th>beersLiked</th>
</tr>
</thead>
<tbody>
<tr>
<td>sue</td>
<td>a</td>
<td>p1</td>
<td>b1</td>
</tr>
<tr>
<td>sue</td>
<td>a</td>
<td>p2</td>
<td>b2</td>
</tr>
</tbody>
</table>

It must also have the same tuples with phones components swapped:

<table>
<thead>
<tr>
<th>name</th>
<th>addr</th>
<th>phones</th>
<th>beersLiked</th>
</tr>
</thead>
<tbody>
<tr>
<td>sue</td>
<td>a</td>
<td>p1</td>
<td>b2</td>
</tr>
<tr>
<td>sue</td>
<td>a</td>
<td>p2</td>
<td>b1</td>
</tr>
</tbody>
</table>

MVD Rules

- Every FD is an MVD: if \( X \rightarrow Y \), then swapping \( Y \)'s between tuples that agree on \( X \) doesn't create new tuples.
- Example, in Drinkers: name \( \rightarrow \rightarrow \) addr.
- Complementation: if \( X \rightarrow \rightarrow Y \), then \( X \rightarrow \rightarrow Z \) where \( Z \) is all attributes not in \( X \) or \( Y \).
- Example: since name \( \rightarrow \rightarrow \) phones holds in Drinkers, so does name \( \rightarrow \rightarrow \) addr beersLiked.

Splitting Doesn't Hold

- Sometimes you need to have several attributes on the right of an MVD.
- For example: Drinkers(name, areaCode, phones, beersLiked, beerManf)

<table>
<thead>
<tr>
<th>name</th>
<th>areaCode</th>
<th>phones</th>
<th>beersLiked</th>
<th>beerManf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leo</td>
<td>773</td>
<td>555-1111</td>
<td>Bud</td>
<td>A.B.</td>
</tr>
<tr>
<td>Leo</td>
<td>773</td>
<td>555-1111</td>
<td>Honkers</td>
<td>G.I.</td>
</tr>
<tr>
<td>Leo</td>
<td>800</td>
<td>555-9999</td>
<td>Bud</td>
<td>A.B.</td>
</tr>
<tr>
<td>Leo</td>
<td>800</td>
<td>555-9999</td>
<td>Honkers</td>
<td>G.I.</td>
</tr>
</tbody>
</table>

- name \( \rightarrow \rightarrow \) areaCode phones holds, but neither name \( \rightarrow \rightarrow \) areaCode nor name \( \rightarrow \rightarrow \) phones do.

Fourth Normal Form (4NF)

- Eliminate redundancy due to multiplicative effect of MVD's.
- Roughly: treat MVD's as FD's for decomposition, but not for finding keys.
- Formally: \( R \) is in Fourth Normal Form if whenever MVD \( X \rightarrow \rightarrow Y \) is nontrivial (\( Y \) is not a subset of \( X \) and \( X \cup Y \) is not all attributes), then \( X \) is a superkey.
  - Remember, \( X \rightarrow Y \) implies \( X \rightarrow \rightarrow Y \), so 4NF is more stringent than BCNF.
  - Decompose \( R \) using 4NF violation \( X \rightarrow \rightarrow Y \) into \( XY \) and \( X \cup (R-Y) \).

Example

Drinkers(name, addr, phones, beersLiked)

- FD: name \( \rightarrow \) addr
- Nontrivial MVD's: name \( \rightarrow \rightarrow \) phones and name \( \rightarrow \rightarrow \) beersLiked.
- Only key: \{name, phones, beersLiked\}
- All three dependencies above violate 4NF.
- Successive decomposition yields 4NF relations:
  - D1(name, addr)
  - D2(name, phones)
  - D3(name, beersLiked)