The Relational Data Model

Functional Dependencies

**Functional Dependencies**

- \( X \rightarrow A \) is an assertion about a relation \( R \) that whenever two tuples of \( R \) agree on all the attributes of \( X \), then they must also agree on the attribute \( A \).
  - Say "\( X \rightarrow A \) holds in \( R \)."
  - Notice convention: \( X, Y, Z \) represent sets of attributes; \( A, B, C, \ldots \) represent single attributes.
  - Convention: no set formers in sets of attributes, just \( ABC \), rather than \( \{A,B,C\} \).

**Example**

- Drinkers(name, addr, beersLiked, manf, favBeer).
- Reasonable FD's to assert:
  1. name -> addr
  2. name -> favBeer
  3. beersLiked -> manf

**Example Data**

<table>
<thead>
<tr>
<th>name</th>
<th>addr</th>
<th>beersLiked</th>
<th>manf</th>
<th>favBeer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Janeway</td>
<td>Voyager</td>
<td>Bud</td>
<td>A.B.</td>
<td>WickedAle</td>
</tr>
<tr>
<td>Janeway</td>
<td>Voyager</td>
<td>WickedAle</td>
<td>A.B.</td>
<td>Pete's</td>
</tr>
<tr>
<td>Spock</td>
<td>Enterprise</td>
<td>Bud</td>
<td>A.B.</td>
<td>Bud</td>
</tr>
</tbody>
</table>

Because name -> addr
Because name -> favBeer
Because beersLiked -> manf

**FD's With Multiple Attributes**

- No need for FD's with > 1 attribute on right.
  - But sometimes convenient to combine FD's as a shorthand.
  - Example: name -> addr and name -> favBeer become name -> addr favBeer
- > 1 attribute on left may be essential.
  - Example: bar beer -> price

**Keys of Relations**

- \( K \) is a **key** for relation \( R \) if:
  1. Set \( K \) functionally determines all attributes of \( R \)
  2. For no proper subset of \( K \) is (1) true.
- If \( K \) satisfies (1), but perhaps not (2), then \( K \) is a **superkey**.
- Note E/R keys have no requirement for **minimality**, as in (2) for relational keys.
Example

Consider relation

\[ \text{Drinkers}(\text{name, addr, beersLiked, manf, favBeer}) \]

\{name, beersLiked\} is a superkey because together these attributes determine all the other attributes.

- \text{name} -> \text{addr}
- \text{beersLiked} -> \text{manf}

Example, Cont.

\{name, beersLiked\} is a key because neither (name) nor (beersLiked) is a superkey.

- \text{name} \rightarrow/=\rightarrow \text{manf}
  (using \(\rightarrow/=\rightarrow\) to mean “does not functionally determine”)
- \text{beersLiked} \rightarrow/=\rightarrow \text{addr}.

In this example, there are no other keys, but lots of superkeys.

- Any superset of \{name, beersLiked\}.

E/R and Relational Keys

- Keys in E/R are properties of entity sets.
- Keys in relations are properties of relations.
- Usually, one tuple corresponds to one entity, so then the ideas are the same.
- But in poor relational designs, one entity set can become several relations, so E/R keys and relational keys are different.

Example Data

<table>
<thead>
<tr>
<th>name</th>
<th>addr</th>
<th>beersLiked</th>
<th>manf</th>
<th>favBeer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Janeway</td>
<td>Voyager</td>
<td>Bud</td>
<td>A.B.</td>
<td>WickedAle</td>
</tr>
<tr>
<td>Janeway</td>
<td>Voyager</td>
<td>WickedAle</td>
<td>Pete’s</td>
<td>WickedAle</td>
</tr>
<tr>
<td>Spock</td>
<td>Enterprise</td>
<td>Bud</td>
<td>A.B.</td>
<td>Bud</td>
</tr>
</tbody>
</table>

Relational key = \text{name} \text{ beersLiked}

But in E/R, \text{name} is a key for Drinkers, and \text{beersLiked} is a key for Beers.

Note: 2 tuples for Janeway entity and 2 tuples for Bud entity.

From Where Do Keys Come?

1. We could simply assert a key \(K\). Then the only \([?]\) FD’s are \(K \rightarrow A\) for all attributes \(A\), and \(K\) turns out to be the only key obtainable from the FD’s.

2. We could assert FD’s and deduce the keys by systematic exploration.

- E/R gives us FD’s from entity-set keys and many-one relationships.

FD’s From “Physics”

- While most FD’s come from E/R keyness and many-one relationships, some are really physical laws.

- Example: “no two courses can meet in the same room at the same time” tells us: \text{hour} \text{ room} \rightarrow \text{course}.

- \[\text{In software engineering, these would be examples of “business rules”}\].]
Inferring FD’s: Motivation

- In order to **design** relation schemas well, we often need to tell what FD’s hold in a relation.

- We are given FD’s $X_1 \rightarrow A_1$, $X_2 \rightarrow A_2$, ..., $X_n \rightarrow A_n$, and we want to know whether an FD $Y \rightarrow B$ must hold in any relation that satisfies the given FD’s.

Example: Transitivity Rule

- If $A \rightarrow B$ and $B \rightarrow C$ hold, surely $A \rightarrow C$ holds as well.

[A Viewpoint of Attributes]

- Whereas one could think of attributes as "labels" on columns, a more mathematical viewpoint is to think of them as **functions**:

  attribute : set-of-tuples $\rightarrow$ set of values

- If $a$ is an attribute and $t$ is a tuple, then $a(t)$ is the value of that attribute in tuple $t$.

[FD Under the Function Interpretation]

- $A_1 A_2 ... A_n \rightarrow B$ means: For any tuples $t$, $t'$ in a relation corresponding to the relation schema:

  *If for all $i$, $A_i(t) = A_i(t')$ then $B(t) = B(t')$.*

Using the Functional Characterization

- Show that $A \rightarrow B$ and $B \rightarrow C$ imply $A \rightarrow C$.

- $A \rightarrow B$ means $\forall t, t' A(t) = A(t')$ implies $B(t) = B(t')$.

- $B \rightarrow C$ means $\forall t, t' B(t) = B(t')$ implies $C(t) = C(t')$.

- Putting these together, we get: $\forall t, t' A(t) = A(t')$ implies $C(t) = C(t')$ so $A \rightarrow C$ as well.

Inference Test

- To test if $Y \rightarrow B$, start assuming two tuples agree in all attributes of $Y$.

- Use the given FD’s to infer that these tuples must also agree in certain other attributes.

- If $B$ is eventually found to be one of these attributes, then $Y \rightarrow B$ is true; otherwise, the two tuples, with any forced equalities form a two-tuple relation that proves $Y \rightarrow B$ does not follow from the given FD’s.
Closure Test

An easier way to test is to compute the closure of $Y$, denoted $Y^*$:

- **Basis:** $Y^* = Y$.
- **Induction:** Look for an FD's left side $X$ that is a subset of the current $Y^*$. If the FD is $X \rightarrow A$, add $A$ to $Y^*$.

Finding All Implied FD’s

- **Motivation:** “normalization,” the process where we break a relation schema into two or more schemas [by projecting it onto the attributes of the new schemas].
- **Example:** $ABCD$ with FD's $AB \rightarrow C$, $C \rightarrow D$, and $D \rightarrow A$.
  - Decompose into $ABC$, $AD$. What FD's hold in $ABC$?
  - Not only $AB \rightarrow C$, but also $C \rightarrow A$!

Simple, Exponential Algorithm

1. For each set of attributes $X$, compute $X^*$.
2. Add $X \rightarrow A$ for all $A$ in $X^* - X$.
3. However, drop $XY \rightarrow A$ whenever we discover $X \rightarrow A$.
   - Because $XY \rightarrow A$ follows from $X \rightarrow A$.
4. Finally, use only FD's involving **projected** attributes.

Basic Idea

- **To know what FD's hold in a projection**, we start with given FD's and find all FD's that follow from given ones.
- Then, **restrict** to those FD's that involve only attributes of the projected schema.

A Few Tricks

- **Never need to compute the closure of the empty set or of the set of all attributes.**
- If we find $X^* = \text{all attributes}$, don’t bother computing the closure of any supersets of $X$. 
**Example**

- ABC with FD’s \( A \rightarrow B \) and \( B \rightarrow C \).
  - Project onto \( AC \):
    - \( A^+ = ABC \); yields \( A \rightarrow B, A \rightarrow C \).
    - We do not need to compute \( AB^+ \) or \( AC^+ \).
    - \( B^+ = BC \); yields \( B \rightarrow C \).
    - \( C^+ = C \); yields nothing.
    - \( BC^+ = BC \); yields nothing.

**Example, Continued**

- Resulting FD’s: \( A \rightarrow B, A \rightarrow C, \) and \( B \rightarrow C \).
- Projection onto \( AC : A \rightarrow C \):
  - Only FD that involves a subset of \( \{A, C\} \).

**A Geometric View of FD’s**

- Imagine the set of all instances of a particular relation.
- That is, all finite sets of tuples that have the proper number of components.
- Each instance is a point in this space.

**Example: \( R(A,B) \)**

- \( R(A,B) = \{(1,2), (3,4)\} \)
- \( A \rightarrow B \)

**An FD is a Subset of Instances**

- For each FD \( X \rightarrow A \) there is a subset of all instances that satisfy the FD.
- We can represent an FD by a region in the space.
- **Trivial FD**: an FD that is represented by the entire space.
  - Example: \( A \rightarrow A \).

**Example: \( A \rightarrow B \) for \( R(A,B) \)**

- \( R(A,B) = \{(1,2), (3,4), (1,3)\} \)
- \( A \rightarrow B \)
Representing Sets of FD’s

- If each FD is a set of relation instances, then a collection of FD’s corresponds to the intersection of those sets.
  - Intersection = all instances that satisfy all of the FD’s.

Example

Implication of FD’s

- If an FD $Y \rightarrow B$ follows from FD’s $X_1 \rightarrow A_1, \ldots, X_n \rightarrow A_n$, then the region in the space of instances for $Y \rightarrow B$ must include the intersection of the regions for the FD’s $X_i \rightarrow A_i$.
  - That is, every instance satisfying all the FD’s $X_i \rightarrow A_i$ surely satisfies $Y \rightarrow B$.
  - But an instance could satisfy $Y \rightarrow B$, yet not be in this intersection.