Schema Refinement and Normal Forms
Database Design Steps

1. Real world
to
2. E/R model
to
3. Relational schema
to
4. **Better relational schema (new!)**
to
5. Relational DBMS

Step 3 to Step 4 is based on a design theory for relations and is called “normalization”; key goal: eliminate redundancy

---

**Movie**

<table>
<thead>
<tr>
<th>title</th>
<th>year</th>
<th>length</th>
<th>starName</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Dancer in the dark”</td>
<td>2000</td>
<td>140'</td>
<td>Bjork</td>
</tr>
<tr>
<td>“Dancer in the dark”</td>
<td>2000</td>
<td>140'</td>
<td>Catherine Deneuve</td>
</tr>
<tr>
<td>“The 39 steps”</td>
<td>1935</td>
<td>86'</td>
<td>Robert Donat</td>
</tr>
<tr>
<td>“The 39 steps”</td>
<td>1978</td>
<td>92'</td>
<td>Robert Powell</td>
</tr>
</tbody>
</table>

**Key?** \{title, year, starName\}

**Problems with this schema?**

- title, year repeated for a movie? Not a problem: both attributes needed to identify movie
- length repeated for a movie? Yes! title and year **determine** length, yet current schema allows for different lengths for the same movie (**redundancy!**), which is a problem
- Also, what if a movie doesn’t have any stars? Where do we store its length?
### 3 Types of Anomalies in Movie Schema

- **Redundancy:** length for a movie stored multiple times
- **Update anomaly:** inconsistencies possible if we update length of a movie
- **Deletion anomaly:** if we delete a star for a movie, we might accidentally lose the movie length as well (and all records of the movie)

**Better design** for this information:
- Movie2(title, year, length)
- Stars(title, year, starName)

Good design with (1) no anomalies and (2) ability to “reconstruct” exactly all original information (how?)

### Functional Dependencies (FDs)

A functional dependency is a special kind of constraint for a relation:
- is based on knowledge of real world
- must always hold for the relation

**Example over Movie table:**
- Functional dependency **title, year → length** holds
- Meaning: any two Movie tuples that agree on both title and year (i.e., on left side of FD) must also agree on length (i.e., on right side of FD)
Functional Dependency: Definition

- A functional dependency $F$ for a relation $R$ consists of one or more attributes $A_1, A_2, \ldots, A_m$ of $R$, followed by symbol $\rightarrow$, followed by one or more attributes $B_1, B_2, \ldots, B_n$ of $R$:
  $$A_1, A_2, \ldots, A_m \rightarrow B_1, B_2, \ldots, B_n$$

- We say that $F$ holds for $R$ if, for every legal instance of $R$, whenever tuples $t$ and $u$ of $R$ agree on all the values of $A_1, A_2, \ldots, A_m$, then $t$ and $u$ must also agree on all the values of $B_1, B_2, \ldots, B_n$.

- For brevity, we often omit the commas that separate attributes (so we write, say, $A_1A_2\ldots A_m \rightarrow B_1B_2\ldots B_n$).

Some Example FDs for Movie

Movie(title, year, length, starName)

- title, year $\rightarrow$ length
- title $\rightarrow$ title
- title, year, starName $\rightarrow$ length
- title, year, starName $\rightarrow$ length, year
- …
Revisiting Definition of “Key” Using FDs

• Consider a relation \( R(C_1, C_2, \ldots, C_t) \) and a set of attributes \( K \subseteq \{C_1, C_2, \ldots, C_t\} \)
• \( K \) is a **key** for \( R \) if and only if:
  1. \( K \rightarrow C_1, C_2, \ldots, C_t \) (i.e., \( K \) “functionally determines” all attributes of relation \( R \))
  2. No strict subset of \( K \) satisfies Condition 1 (i.e., \( K \) is minimal)
• In general, if a set of attributes \( K' \subseteq \{C_1, C_2, \ldots, C_t\} \) is such that \( K' \rightarrow C_1, C_2, \ldots, C_t \), then \( K' \) is a **superkey** for \( R \)

  If additionally \( K' \) has the minimality property (Condition 2 above), then \( K' \) is also a key (so every key is a superkey but not the other way around)

Characterizing Functional Dependencies

Consider a functional dependency \( X \rightarrow A \), where \( X \) and \( A \) are sets of attributes of a relation

• \( X \rightarrow A \) is **trivial** if \( A \subseteq X \)
  • title, length \( \rightarrow \) title
  • length, year \( \rightarrow \) length, year
• \( X \rightarrow A \) is **nontrivial** if \( A \not\subseteq X \)
  • title, year \( \rightarrow \) title, length
• \( X \rightarrow A \) is **completely nontrivial** if \( X \cap A = \emptyset \)
  • title, year \( \rightarrow \) length

Most of the time we are interested in completely nontrivial FDs
Reasoning About FDs

• Given some FDs, we can infer additional FDs

• **“Decomposition” Rule:** If \( X \rightarrow B_1, B_2, \ldots, B_n \) holds, then so do \( X \rightarrow B_1, X \rightarrow B_2, \ldots, \) and \( X \rightarrow B_n \)
  
  • Employees(ssn, name, age)
  
  • If we are told \( ssn \rightarrow name, age \) holds, then we know \( ssn \rightarrow name \) and \( ssn \rightarrow age \) also hold
  
  • Can we “split” the left side also? **No!** title, year \( \rightarrow \) length holds on Movie, yet title \( \rightarrow \) length and year \( \rightarrow \) length do **not** hold

• **“Union” Rule:** If \( X \rightarrow B_1, X \rightarrow B_2, \ldots, \) and \( X \rightarrow B_n \) hold, then so does \( X \rightarrow B_1, B_2, \ldots, B_n \)
  
  • If \( ssn \rightarrow name \) and \( ssn \rightarrow age \) hold, so does \( ssn \rightarrow name, age \)

---

Reasoning About FDs

• FD F is **“implied”** by a set of FDs S if F holds whenever all FDs in S hold

• The **closure of S, S⁺**, is the set of all FDs implied by S

• To compute the closure of a set of FDs, we can use **Armstrong’s axioms:**
  
  • **Reflexivity:** If \( Y \subseteq X \), then \( X \rightarrow Y \)
  
  • **Augmentation:** If \( X \rightarrow Y \) holds, then \( X \cup Z \rightarrow Y \cup Z \) also holds for any attribute set \( Z \)
  
  • **Transitivity:** If \( X \rightarrow Y \) and \( Y \rightarrow Z \) hold, then so does \( X \rightarrow Z \)

• Armstrong’s axioms are a **sound** and **complete** set of inference rules for FDs
  
  • The “decomposition” and “union” rules follow from Armstrong’s axioms. in particular
Using Armstrong's Axioms: An Example

R(A, B, C, D, E) is a relation with given FDs:
(1) A \rightarrow B, (2) B \rightarrow C, and (3) CD \rightarrow E

Question: Does AD \rightarrow E hold for R given that (1), (2), and (3) hold?

Answer: Use Armstrong's axioms to prove that it does, as follows:

• From (1) and (2), and transitivity: (4) A \rightarrow C
• From (4) and augmentation: (5) AD \rightarrow CD
• From (5) and (3), and transitivity: AD \rightarrow E

“Closure” of Set of Attributes

• As an alternative to the axioms for reasoning about what FDs hold, we can use the “closure” of a set of attributes

• Consider a relation R, a set of FDs S for R, and a set of attributes \{A_1, \ldots, A_k\} of R. Then the closure of \{A_1, \ldots, A_k\}, denoted \{A_1, \ldots, A_k\}^+, consists of all attributes B in R such that A_1, A_2, \ldots, A_k \rightarrow B
Closure of Set of Attributes: An Algorithm

To compute the closure of \( \{A_1, \ldots, A_k\} \), \( \{A_1, \ldots, A_k\}^+ \), for a relation R with respect to a set of FDs S for R:

- Start with \( \{A_1, \ldots, A_k\} \)
- Repeat until no change:
  - If current set of attributes includes the full left side of a FD in S, then add all the right side attributes of the FD to the set
- Return the set of attributes

Using Closure of Set of Attributes: Example Revisited

R(A, B, C, D, E) is a relation with given FDs:
(1) A \( \rightarrow \) B, (2) B \( \rightarrow \) C, and (3) CD \( \rightarrow \) E

Question: Does AD \( \rightarrow \) E hold for R given that (1), (2), and (3) hold?

Answer: Compute \( \{A, D\}^+ \) and check if E is in the closure

- Start with closure=\( \{A, D\} \)
- closure includes left side of (1) (i.e., attribute A), so add right side (i.e., attribute B) to closure: closure=\( \{A, D, B\} \)
- closure includes left side of (2) (i.e., attribute B), so add C to closure: closure=\( \{A, D, B, C\} \)
- closure includes left side of (3) (i.e., attributes C and D), so add E to closure: closure=\( \{A, D, B, C, E\} \)
- No more changes possible, so return \( \{A, D\}^+ = \{A, D, B, C, E\} \)
- Because E is in \( \{A, D\}^+ \), it follows that AD \( \rightarrow \) E holds
Identifying All Keys for a Relation

Consider a relation $R(A_1, \ldots, A_n)$ and a given set of functional dependencies $S$ that hold over $R$. **How do we determine all keys for $R$?**

**Task:** Identify every set $K$ of attributes of $R$ such that:
1. $K \rightarrow$ all attributes of $R$
2. $K$ is minimal with such property (i.e., no strict subset of $K$ functionally determines all attributes of $R$)

**Algorithm for Finding All Keys for Relation $R$**

(note it is exponential in the number of attributes $n$ of the relation)

For $i = 1$ to $n$:

- Consider each set of attributes $C_i$ with exactly $i$ attributes of $R$
- If $C_i$ includes a (previously found) key, then ignore $C_i$ ($C_i$ could not satisfy minimality)
- Otherwise, compute the closure of $C_i$: if $C_i^+ = \{A_1, \ldots, A_n\}$, then $C_i$ is a key for $R$

Is a Relation Schema “Good”?  

- “Normal forms” determine which relation schemas are “good,” in the sense of avoiding certain kinds of redundancy

- **Boyce-Codd Normal Form (BCNF):** Consider a relation $R$ and a set of functional dependencies $S$ for $R$

  $R$ is in BCNF with respect to FD set $S$ if for every **nontrivial** FD $X \rightarrow A$ that holds over $R$, $X$ contains a key (i.e., $X$ is a superkey)
Why is violating BCNF bad?

Employees(ssn, name, did, rank, sal)

Given FDs:
- ssn → name, did, rank
- did, rank → sal

First, we can use the algorithm above to identify all keys of the relation: {ssn} is the only key

{ssn}⁺ includes all attributes of the relation and, of course, {ssn} is minimal; furthermore, no other attribute set satisfies these two conditions

Then Employees is not in BCNF: did, rank → sal holds, it is nontrivial, and its left side does not include ssn

Why is this a problem? Value of sal for a did-rank pair is potentially stored many times in relation...

Another Example

Movie(title, year, length, starName)

Given FD: title, year → length

- Only key of the relation is {title, year, starName}
- title, year → length holds for Movie, it is nontrivial, and its left side doesn’t include a key (it’s missing starName)
- Hence Movie is not in BCNF
Handling BCNF Violations

As we will see, when a relation is not in Boyce-Codd Normal Form, we can decompose it into “smaller” relations to avoid redundancy:

- Split Employees(ssn, name, did, rank, sal) into:
  - Employees2(ssn, name, did, rank) and
  - Salaries(did, rank, sal)

- Split Movie(title, year, length, starName) into:
  - Movie2(title, year, length)
  - Stars(title, year, starName)