8.2 Functional Dependencies

Proceeding

- DB schema + functional dependencies
- decomposition of the given database schema into an equivalent schema without redundancy and anomalies (**normalization**)
- integrity constraints: conditions for the permitted instances of a database schema
- A **functional dependency** (FD) is a special integrity constraint.
- In the following let $R$ be the relation schema of a relation $R$.
- definition of functional dependency
  
  Let $A, B \subseteq R$. $B$ is **functionally dependent** on $A$, or $A$ determines $B$ functionally (uniquely), written $A \rightarrow B$, iff to each value in $A$ exactly one value in $B$ belongs:
  
  $$A \rightarrow B \iff \forall t_1, t_2 \in R(R) : t_1[A] = t_2[A] \Rightarrow t_1[B] = t_2[B]$$

  for all possible relations $R$ over $R$.
- A functional dependency depends on the **semantics** of the **schema**, and not on the instance of a relation.
example:

- `supplier(sname, saddr, product, price)`
- functional dependencies:
  + `{sname} → {saddr}`
    A supplier’s name determines uniquely the supplier’s address.
  + `{sname, product} → {price}`
    The key `{sname, product}` determines uniquely the price.
  + `{sname} → {sname}, {sname, product} → {product} are trivial.
  + `{sname, product} → {saddr} is partial

A dependency is called **trivial** if \( B \subseteq A \) holds.

A functional dependency \( X \rightarrow Y \) is called **full**, if there is no proper subset \( Z \subset X \) so that \( Z \rightarrow Y \) holds. If such a subset exists, \( X \rightarrow Y \) is called **partial** dependency.

Let \( X, Y \subset R \), and let \( X \rightarrow Y \). Let \( A \in R \) be an attribute with \( A \not\in X, A \not\in Y \), and let \( Y \rightarrow \{A\} \). Then \( A \) is **transitively dependent** on \( X \), i.e., \( X \rightarrow \{A\} \).
Checking the preservation of a functional dependency

- Alternative characterization of a FD $A \rightarrow B$
  
  Let $A = \{A_1, \ldots, A_n\}$, and let $\text{dom}(A) = \text{dom}(A_1) \times \ldots \times \text{dom}(A_n)$.

  The FD $A \rightarrow B$ holds on $R$ if $\forall v \in \text{dom}(A) : |\pi_B(\sigma_{A=v}(R))| \leq 1$

  ($A = v$ stands for $A_1 = v_1 \land \ldots \land A_n = v_n$)

- This leads to a simple algorithm which computes whether a given relation $R$ satisfies the FD $A \rightarrow B$:

  **algorithm** FDPreservation($R, A \rightarrow B$)

  // input: relation $R$ and FD $A \rightarrow B$

  // output: true, if $A \rightarrow B$ holds on $R$; false otherwise

  sort $R$ with respect to $A$-values

  if all groups consisting of tuples with equal $A$-values also have equal $B$-values **then**
   
  return true

  **else**
   
  return false

end.
Computation of FDs

- Goal: Compute for a given set $F$ of FDs all **logically implied** FDs.

- Let $F^+$ be the set of all FDs that can be logically implied from the FDs in $F$. $F^+$ is called the **closure** of $F$.

- Let $R$ be a relation schema, $F$ a set of FDs and $A, B, C \subseteq R$.

  The following **inference rules** are used to compute $F^+$ (Armstrong’s axioms):
  - **reflexivity rule**: Let $B \subseteq A$. Then always $A \rightarrow B$ (special case: $A \rightarrow A$) holds.
  - **augmentation rule**: If $A \rightarrow B$ holds, then also $A \cup C \rightarrow B \cup C$ holds.
  - **transitivity rule**: If $A \rightarrow B$ and $B \rightarrow C$ holds, then also $A \rightarrow C$ holds.

- It can be formally shown that these rules are **sound** and **complete**.
  - **soundness**: Inferred FDs hold for all relations of this schema.
  - **completeness**: All valid FDs in $F^+$ can be logically implied with these rules.


Although Armstrong’s axioms are complete, it is comfortable to add three further inference rules:

- **union rule**: If \( A \rightarrow B \) and \( A \rightarrow C \) holds, then also \( A \rightarrow B \cup C \) holds.
- **decomposition rule**: If \( A \rightarrow B \cup C \) holds, then also \( A \rightarrow B \) and \( A \rightarrow C \) holds.
- **pseudotransitivity rule**: If \( A \rightarrow B \) and \( B \cup C \rightarrow D \) holds, then also \( A \cup C \rightarrow D \) holds.

**example:**

- **supplier** relation with the schema \( \text{supplier}(\text{sname}, \text{saddr}, \text{product}, \text{price}) \)
  - Valid FDs: \( \{\text{sname}\} \rightarrow \{\text{saddr}\} \), \( \{\text{sname}, \text{product}\} \rightarrow \{\text{price}\} \), \( \{\text{sname}\} \rightarrow \{\text{sname}\} \), \( \{\text{sname}, \text{product}\} \rightarrow \{\text{product}\} \)
  - It is to be shown: \( \{\text{sname}, \text{product}\} \rightarrow \{\text{saddr}\} \) is also satisfied.

  We have: \( \{\text{sname}\} \rightarrow \{\text{saddr}\} \).

  Due to the augmentation rule we obtain: \( \{\text{sname}, \text{product}\} \rightarrow \{\text{saddr}, \text{product}\} \).

  Due to the decomposition rule we hence obtain: \( \{\text{sname}, \text{product}\} \rightarrow \{\text{saddr}\} \).
computing the closure $F^+$

$F^+ = F$

repeat

for each functional dependency $f$ in $F^+$ do

apply reflexivity and augmentation rules to $F^+$
add the resulting functional dependencies to $F^+$

od;

for each pair of functional dependencies $f_1$ and $f_2$ in $F^+$ do

if $f_1$ and $f_2$ can be combined using transitivity then

add the resulting functional dependency to $F^+$

fi

od;
until $F^+$ does not change any further