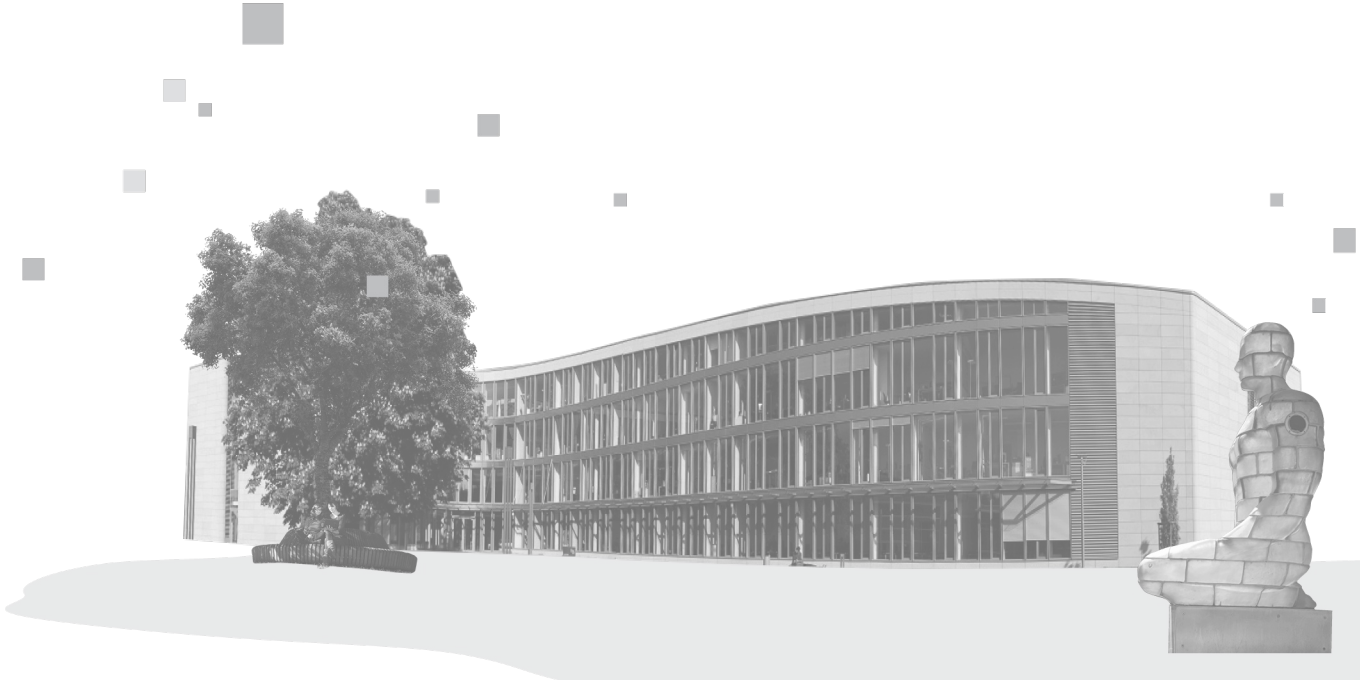
Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Advanced Data Profiling

FD Membership Test

WS 2023/2024
Daniel Lindner



Design IT.
Create Knowledge.

www.hpi.de

Overview

1. Recap: Functional Dependency
2. Axioms
3. Intuition: Derivation Trees
4. Membership Test

Recap: Functional Dependency [3, 4]

- $t[X]$ is the projection of tuple t to attributes X

> *Given a relational instance r of schema R, the **functional dependency** $X \rightarrow Y$ with attribute set $X \subseteq R$ and attribute set $Y \subseteq R$ is valid in r iff $\forall\, t_i, t_j \in r : t_i[X] = t_i[X] \Longrightarrow t_i[Y] = t_i[X]$.*

- "All tuples with the same values for X also have the same values for Y."
- some (basic) examples:
  - $\{zip\} \rightarrow \{city\}$
  - $\{month\} \rightarrow \{meteorological\ season\}$
  - $\{package,\ weight,\ size\} \rightarrow \{shipping\ cost\}$
  - $X \rightarrow R \setminus X$ if X is a UCC (all primary keys)

Armstrong‘s Axioms [1]

1. Reflexivity: $X \rightarrow X$
   “Each attribute depends on itself.”
2. Transitivity: $X \rightarrow Y \ \wedge Y \rightarrow Z \Rightarrow X \rightarrow Z$
3. Augmentation: $X \rightarrow Y \Rightarrow X \ \cup A \rightarrow Y \setminus B$ (where A, B can be $\emptyset$)
   “We can add LHS attributes and remove RHS attributes.”
4. Union: $X \rightarrow Y \wedge A \rightarrow B \Rightarrow X \cup A \rightarrow Y \cup B$
   “We can merge valid FDs.”

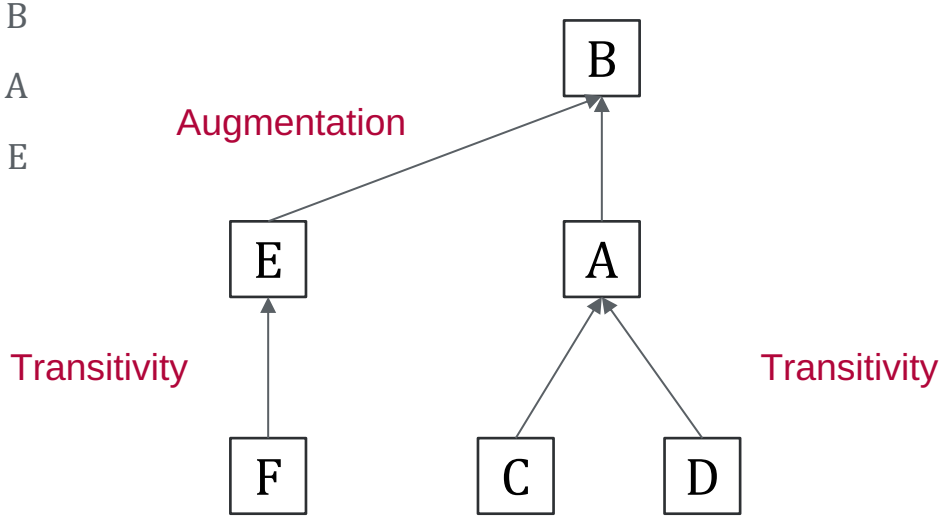
Intuition: Derivation Trees [2]

- Build a tree with determinant as leaf, dependent as root
- From now on, we omit brackets for sets: $\{A, B\} \rightarrow \{C\}$ is written as $A, B \rightarrow C$

$A \rightarrow B$

$C, D \rightarrow A$

$F \rightarrow E$



- Derived $C, D, F \rightarrow B$ from $A \rightarrow B$
- Tree represents all FDs from children to (parents of …) parents

Membership Test – Idea [2]

- Given set of FDs $\mathcal{G}$, is FD f: $\mathrm{A}, \mathrm{B} \rightarrow \mathrm{X}, \mathrm{Y}$ valid? $\Leftrightarrow f \in \mathcal{G}^*$?
- Building all derivation trees from A, B possible, but not efficient
- Based on two intuitions (proof in [2])
  - All nodes in a tree are reachable from (transitive) children
  - Valid trees can be merged and augmented
- → Construct all derivation trees at once

Membership Test – Algorithm [2]

- Given set of FDs $\mathcal{G}$, is FD f: $\mathrm{A}, \mathrm{B} \rightarrow \mathrm{X}, \mathrm{Y}$ valid?

```
reachable_dependants = f.LHS;  Reflexivity
found_new_dependant = true;

while (found_new_dependant):
    found_new_dependant = false;
    foreach g ∈ G:
        if (g.LHS ⊆ reachable_dependants ∧ g.RHS ⊄ reachable_dependants):
                                                          Transitivity
            all_dependants ∪ g.RHS;   Augmentation, Union
            found_new_dependant = true;

return f.RHS ⊆ reachable_dependants;
```

Test **all** LHS attributes for **all** FDs multiple times!

- Complexity $O(|\mathcal{G}| \cdot m)$, where m is the number of attributes

Membership Test – Improved Algorithm [2]

- Given set of FDs $\mathcal{G}$, is FD f: $\mathrm{A}, \mathrm{B} \rightarrow \mathrm{X}, \mathrm{Y}$ valid?

```
reachable_dependants = f.LHS;
found_new_dependant = true;
fds_per_attribute = Map<attribute → FD[]>;
remaining_attributes = Queue<attribute>(f.LHS);

foreach g ∈ 𝒢:
    g.required_attributes = g.LHS.size();
    foreach a ∈ g.LHS:
        fds_per_attribute[a].append(g);

while (!remaining_attributes.empty()):
    a = remaining_attributes.pop();
    foreach g ∈ fds_per_attribute[a]:
        g.required_attributes--;
        if (g.required_attributes == 0):
            foreach dependent ∈ g.RHS:
                if (dependent ∉ reachable_dependants)
                    all_dependants ∪ {dependent};
                    remaining_attributes.push(dependent);

return f.RHS ⊆ reachable_dependants;
```

Test each reachable LHS attribute once!

References

[1] William W. Armstrong: **Dependency Structures of Data Base Relationships**. IFIP Congress 1974.

[2] Philip A. Bernstein and Catriel Beeri: **An Algorithmic Approach to Normalization of Relational Database Schemas.** Technical Report CSRG-73. Computer Systems Research Group, University of Toronto, 1976.

[3] Edgar F. Codd: **Further Normalization of the Data Base Relational Model.** Research Report RJ909. IBM, 1971.

[4] Jeffrey D. Ullman: **Principles of Database and Knowledge-Base Systems, Volume I.** Principles of computer science series 14, Computer Science Press 1988.