Functional Dependency

by

Dr. Pratik Roy

Definition of Functional Dependency

- Functional dependency is a constraint between two sets of attributes.
- Suppose that our relational database schema R has n attributes A_1, A_2, ..., A_n; R = {A_1, A_2, ... , A_n}.
- A functional dependency, denoted by X → Y, between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation state r of R. The constraint is that, for any two tuples t_1 and t_2 in r that have $t_1[X] = t_2[X]$, they must also have $t_1[Y] = t_2[Y]$.
- Values of Y component of a tuple in r depend on, or are determined by, values of X component.
- Values of X component of a tuple uniquely (or functionally) determine values of Y component.
- There is a functional dependency from X to Y, or that Y is functionally dependent on X.

Example

roll_no	name	dept_name	dept_building
42	abc	CO	A4
43	pqr	IT	A3
44	xyz	CO	A4
45	xyz	IT	A3
46	mno	EC	B2
47	jkl	ME	B2

Some valid functional dependencies:

- roll_no → { name, dept_name, dept_building }, Here, roll_no can determine values of fields name, dept_name and dept_building, hence a valid Functional dependency
- roll_no → dept_name, Since, roll_no can determine whole set of {name, dept_name, dept_building}, it can determine its subset dept_name also.

Example

- dept_name → dept_building, Dept_name can identify the dept_building accurately, since departments with different dept_name will also have a different dept_building
- More valid functional dependencies: roll_no → name, {roll_no, name} → {dept_name, dept_building}, etc.

Some invalid functional dependencies:

- name → dept_name, Students with same name can have different dept_name, hence this is not a valid functional dependency.
- dept_building → dept_name, There can be multiple departments in the same building, For example, in the above table departments ME and EC are in the same building B2, hence dept_building → dept_name is an invalid functional dependency.
- More invalid functional dependencies: name → roll_no, {name, dept_name} → roll_no, dept_building → roll_no, etc.

Superkey

- Given r(R), a subset K of R is a superkey of r(R) if, in any legal instance of r(R), for all pairs t_1 and t_2 of tuples in the instance of r if $t_1 \neq t_2$, then $t_1[K] \neq t_2[K]$.
- No two tuples in any legal instance of relation r(R) may have same value on attribute set K.
- If no two tuples in r have the same value on K, then a K-value uniquely identifies a tuple in r.
- K is a superkey for r(R) if functional dependency $K \rightarrow R$ holds on r(R).
- K is a superkey if, for every legal instance of r(R), for every pair of tuples t_1 and t_2 from the instance, whenever $t_1[K] = t_2[K]$, it is also the case that $t_1[R] = t_2[R]$ (i.e., $t_1 = t_2$).

Functional Dependency

- If X→Y in R, this does not say whether or not Y→X in R.

A	B	C	D
a1	b1	c1	d1
a1	b2	c1	D2
a2	b2	c2	d2
a2	b3	c2	d3
a3	b3	c2	d4

- $A \rightarrow C$ is satisfied, $C \rightarrow A$ is not satisfied.

Trivial Functional Dependency

- Some functional dependencies are said to be trivial because they are satisfied by all relations.
- **Example**
  - A→A is satisfied by all relations involving attribute A.
  - AB → A is satisfied by all relations involving attribute A.
- If X → Y and Y is the subset of X, then it is called trivial functional dependency.

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

{roll_no, name} → name
roll_no → roll_no

Non-trivial Functional Dependenc

- If X → Y and Y is not a subset of X, then it is called Non-trivial functional dependency.
- **Example**

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

roll_no → name
{roll_no, name} → age

Properties of functional dependencies

❑Armstrong's axioms

- **Reflexivity rule:** If α is a set of attributes and $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ holds.
- **Augmentation rule:** If $\alpha \rightarrow \beta$ holds and γ is a set of attributes, then $\gamma\alpha \rightarrow \gamma\beta$ holds.
- **Transitivity rule:** If $\alpha \rightarrow \beta$ holds and $\beta \rightarrow \gamma$ holds, then $\alpha \rightarrow \gamma$ holds.

❑Additional rules

- **Union rule:** If $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds.
- **Decomposition rule:** If $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds.
- **Pseudotransitivity rule:** If $\alpha \rightarrow \beta$ holds and $\gamma\beta \rightarrow \delta$ holds, then $\alpha\gamma \rightarrow \delta$ holds.

Closure

- We denote by F the set of functional dependencies that are specified on relation schema R.
- **Closure (F^+):** Set of all functional dependencies that can be inferred given the set F.