Functional Dependency

Functional Dependency

A "good" database schema should not lead to *update anomalies*.

- update anomalies,

- functional dependencies,

- Armstrong Axioms,

- closures.

Update Anomalies

Redundancy in a database means storing a piece of data more than once.

Redundancy is often useful for efficiency and semantic reasons, but creates the potential for consistency problems.

A poor *redundancy control* may cause update anomalies.

Consider the example relation below (adapted from "An Introduction to Database Systems" by Desai):

STUDENTS					
Name	Course	Phone_no	Major	Prof	Grade
Jones	353	237-4539	Comp Sci	Smith	A
Ng	329	427-7390	Chemistry	Turner	B
Jones	328	237-4539	Comp Sci	Clark	B
Martin	456	388-5183	Physics	James	A
Dulles	293	371-6259	Decision Sci	Cook	C
Duke	491	823-7293	Mathematics	Lamb	B
Duke	356	823-7293	Mathematics	Bond	UN
Jones	492	237- 4539	Comp Sci	Cross	UN
~~Baxter~~	~~379~~	~~839-0827~~	~~English~~	~~Broes~~	~~C~~

Modification anomalies: e.g. Jones's phone number appears 3 times. When a phone number is changed, it must be changed in all 3 places, or the data will be inconsistent.

Update Anomalies

Insertion anomalies:

• If Jones enrolls in another course, and a different phone number is entered, again the data will be inconsistent.

• Also, if the only way that the association between course and professor is stored in this relation, we can only enter the association when someone enrolls in the course.

Deletion anomalies: If the last student in a course is deleted, the association between professor and course is lost.

Functional dependencies

A function f from S_1 to S_2 has the property

$$if \ x, y \ \in S_1 \ and \ x = y, then \ f(x) = f(y) \ .$$

A generalization of keys to avoid design flaws violating the above rule.

Let X and Y be sets of attributes in R.

X (*functionally*) determines Y, $X \rightarrow Y$, iff $t_1[X] = t_2[X]$ implies $t_1[Y] = t_2[Y]$.

i.e., f (t(X)) = t [Y]

We also say $X \rightarrow Y$ is a *functional* dependency, and that Y is *functionally* dependent on X.

X is called the *left side*, Y the *right side* of the dependency.

Examples

• For every Name, there is a unique Phone_no and Major, assume Name is unique;

• For every Course, there is a unique Prof;

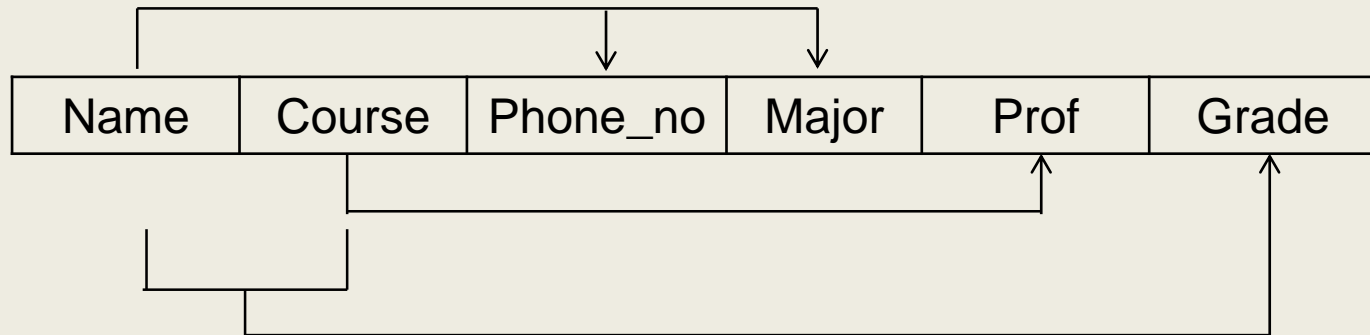• For every Name and Course, there is a unique

Grade.

In this example:

$$\{Name\} \rightarrow \{Phone\_no , Major\}$$

$$\{Course\} \rightarrow \{Prof\}$$

$$\{Name , Course\} \rightarrow \{Grade\}$$

We can also show these in a diagram like this one:

Name	Course	Phone_no	Major	Prof	Grade

Notice that other FD's follow from these:

$$\{Name\} \rightarrow \{Major\}$$

$$\{Course , Grade\} \rightarrow \{Prof , Grade\}$$

Functional dependencies

Let F be a set of FD's.

Definition 1: $X \rightarrow Y$ is inferred from F (or that F infers $X \rightarrow Y$), written in

$$F \models X \rightarrow Y$$

if any relation instance satisfying F must also satisfy $X \rightarrow Y$.

Impossible to list every relation to verify if $X \rightarrow Y$ is inferred from F.

A set ρ of derivation rules are required, such that:

a $X \rightarrow Y$ is inferred from F according to Definition 1 iff it can be

derived using ρ.

Armstrong's axioms (1974)

Notation: If X and Y are sets of attributes, we write XY for their union.

e.g. X = {A, B}, Y = {B, C}, XY = {A, B, C}

F1 (Reflexivity) If $X \supseteq Y$ then $X \rightarrow Y$.

F2 (Augmentation) $\{X \rightarrow Y\} \models XZ \rightarrow YZ$.

F3 (Transitivity) $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$.

F4 (Additivity) $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$.

F5 (Projectivity) $\{X \rightarrow YZ\} \models X \rightarrow Y$.

F6 (Pseudotransitivity) $\{X \rightarrow Y, YZ \rightarrow W\} \models XZ \rightarrow W$.

Example: Given F = $\{A \to B, A \to C, BC \to D\}$, derive $A \to D$:

1. $A \to B$ (given)

2. $A \to C$ (given)

3. $A \to BC$ (by F4, from 1 and 2)

4. $BC \to D$ (given)

5. $A \to D$ (by F3, from 3 and 4)

F4 (Additivity) $\{X \to Y, X \to Z\} \models X \to Y Z$.

F5 (Projectivity) $\{X \to Y Z\} \models X \to Y$.

F6 (Pseudotransitivity) $\{X \to Y, Y Z \to W\} \models XZ \to W$.

In fact, F4, F5, and F6 can be derived from F1-F3.

Example: Prove $\{X \to Y, X \to Z\} \models X \to Y Z$.

1) $X \to Y$ is given.

2) $XX \to XY$ (by F2)*; that is,* $X \to XY$

3) $X \to Z$ is given.

4) $X Y \to Y Z$ (by F2)

5) $X \to Y Z$ (by F3, 2) and 4))

Armstrong's axioms

We can prove that Armstrong's axioms are sound and complete:

Sound: if F derives $A \to B$ by using Armstrong's axioms, then F $\models A \to B$ by Definition 1.

Complete: if F $\models M \to N$ by Definition 1, then F derives $M \to N$ by using Armstrong's axioms.

Algorithm to Check a FD

Given F, how do we check if $X \rightarrow Y$ is in F^+?

F^+ denotes the smallest set of FD's that

- contains F, and

- is *closed* under Armstrong's axioms.

F^+ is the *closure* of F.

F = { A → B, B → C, A → C }

F$^+$ = {AB -> A, AB -> B, AB -> C, AC -> A, AC -> B, AC -> C, ABC -> A, ABC -> B, ABC -> C, AB -> AB, AB -> BC, AB -> AC, …….}

F$^+$ always has an exponential size regarding |F|.

Too expensive to compute F^+ to verify a membership.

Instead we can compute the *closure* X^+ of X under F, X^+ is the largest set of attributes functionally determined by X.

It can be proven (using additivity) that

S1:
$$X^+ = \cup_{\forall X \rightarrow A \in F+} A.$$

S2: $\quad X \rightarrow Y \in F^+$ iff (if and only if) $Y \subseteq X^+$.

Example:

F = { A → B, BC → D, A → C }, compute $\{A\}^+$

1st scan of F:
$X^+ := \{A\}$
$X^+ := \{A, B\}$
$X^+ := \{A, B, C\}$

2nd scan of F:
$X^+ := \{A, B, C, D \}$

3rd scan of F: no change, therefore the algorithm terminates.

$\{A\}^+ := \{A, B, C, D \}$

Algorithm to compute X⁺

$X^+ := X$;

change := true;

while change do

        begin

        change := false;

        for each FD W → Z in *F* do

                begin

                if $(W \subseteq X^+)$ and $(Z \not\subseteq X^+)$ then do

                        begin

                        $X^+ := X^+ \cup Z$;

                        change := true;

                        end

                end

        end

Algorithm to Compute a Candidate Key

Given a relational schema R and a set F of functional dependencies on R.

A key X of R must have the property that $X^+ = R$.

Algorithm to compute a candidate key

Step 1: Assign X a superkey in F.

Step 2: Iteratively remove attributes from X while retaining the property $X^+ = R$ till no reduction on X.

The remaining X is a key.

Example:

R = {A, B, C, D} and F = { A → B, BC → D, A → C }

X = {A, B, C} if the left hand side of F is a super key.

A cannot be removed because $\{BC\}^+ = \{B, C, D\} \neq R$

B can be removed because $\{AC\}^+ = \{A, B, C, D\} = R$
⟶ X = { A, C}

C can be further removed because $\{A\}^+ = \{A, B, C, D\}$
⟶ X = {A}