FUNCTIONAL DEPENDENCIES

Examples: PERSON Relation

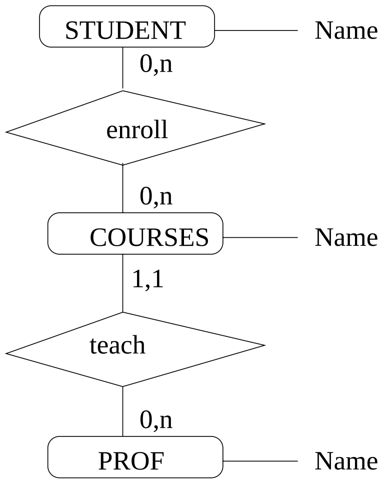
PERSON	SIN	NAME	CITY
	123	Laurent	Toronto
	324	Bill	Toronto
	574	Bill	Montreal

What can we say about Person table?

- "If I know the sin number I know the name"
- `SIN` attribute determines `NAME` attribute.
- Attribute `NAME` *functionally depends* on attribute `SIN`
- Warning: Knowing the `NAME` does not imply the SIN knowledge: `NAME` $\not\rightarrow$ `SIM`

NOTATION: `SIN` $\rightarrow$ `NAME`

COURSES Relation



COURSE	NAME	PROF	STUDENT
	Database	MIGNET	SMITH
	Database	MIGNET	BILL
	Database	MIGNET	SMITH
	Math	HARDIN	GEORGE

- “A Course has only one Professor”
- NAME $\rightarrow$ PROF
- 2 tuples that have the same value for NAME have the same value for PROF

Key Example

`PERSON (SIN, LastName, FirstName, Address)`

SIN $\rightarrow$ LastName
SIN $\rightarrow$ FirstName
SIN $\rightarrow$ Address

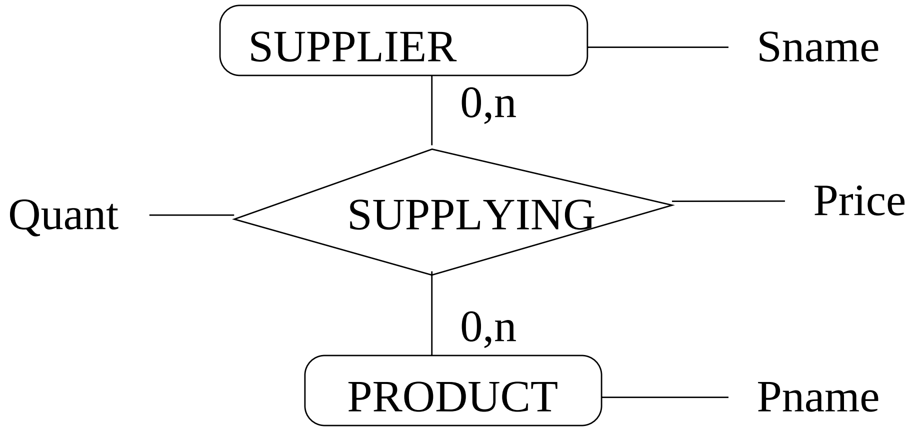
- If we know the SIN value, we know all the attributes.
- 2 tuples sharing the same SIN are *identical*
- SIN *identifies* a tuple.
- SIN is a *key*

SIN LastName $\rightarrow$ FirstName
SIN LastName is a *superkey* (LastName is redundant)

SUPPLYING Relation

`SUPP (SNAME, PNAME, QUANT, PRICE)`

SNAME PNAME $\rightarrow$ QUANT
SNAME PNAME $\rightarrow$ PRICE
SNAME PNAME is a key.



ADDRESS Relation

`ADDRESS (STREET, NUMBER, CITY, ZIPCODE)`

Simplifying Hypothesis

- Several zipcode for a city: Toronto = M5B3F4, M5S2E4, ...
- One city for a zipcode: Mxxxxx = Toronto
- An address (street + number + city) belongs only to one zipcode: 4 St George Street, Toronto = M5S 2E4

Number Street City $\rightarrow$ Zipcode
Zipcode $\rightarrow$ City

Keys of the relation ADDRESS: (Street Number City) and (Number Street Zipcode)

Definitions

a) Functional Dependencies

Let `R(U)` a relational schema, `r` one relation of the schema `R`, $X \subset U$, $Y \subset U$ two attribute sub-set of `R`. The *functional dependencies*

$$X \rightarrow Y$$

is true in `r`, iff (if and only if) for every tuples of `r` that share the same value for all attributes of `X`, they also share the same value for all attributes of `Y`.

Example:

```
ADDRESS(STREET, NUMBER, CITY, ZIPCODE)
X = ZipCode
Y = City
```

b) SuperKey

Let $\mathtt{R(U)}$ a schema and $\mathtt{X} \subset \mathtt{U}$ a attribute subset.
$\mathtt{X}$ is a superkey *superkey* of $\mathtt{r}$ of the schema $\mathtt{R}$, if $\mathtt{X} \rightarrow \mathtt{U}$.

Example:

$\mathtt{SIN\ LASTNAME} \rightarrow \mathtt{SIN\ LASTNAME\ FIRSTNAME\ ADDRESS}$

$\mathtt{SIN\ LASTNAME}$ is a superkey

c) KEY

$\mathtt{X}$ is a key, if:

1. $\mathtt{X}$ is a superkey: $\mathtt{X} \rightarrow \mathtt{U}$
2. it does not exist $\mathtt{Y} \subset \mathtt{X}$, such that $\mathtt{Y} \rightarrow \mathtt{U}$

Example:

$\mathtt{STREET\ NUMBER\ CITY} \rightarrow \mathtt{STREET\ NUMBER\ CITY\ ZIPCODE}$

$\mathtt{STREET\ NUMBER\ CITY}$ is a key (why?)

Finding a Key: Example

COURSE(Name,Hour,Room,Prof)

$\mathcal{F} = \texttt{N} \rightarrow \texttt{HR},\ \texttt{HR} \rightarrow \texttt{P}$

We can proof from $\mathcal{F}$ that if we know the name of the course, we also know the name of the professor. (`Name` is a key):

1. $\texttt{N} \rightarrow \texttt{HR}$: two tuples that share the name of the course share also the hour and the room value.
2. $\texttt{HR} \rightarrow \texttt{P}$: two tuples that share the hour and the room share also the name of the professor.
3. FD 1 and 2 implie that two tuples which have the same name for the course have also the same professor name: $\texttt{N} \rightarrow \texttt{P}$

We can define some properties and rules on the FD which permit to deduct others FDs.

Functional Dependencies Properties

(Armstrong's Axioms)

1) Reflexivity

If $X \subseteq Y$ then $Y \rightarrow X$ (for every attribute subsets X et Y)

Example:

NAME CITY $\rightarrow$ NAME

Trivial: "Two persons who have the same name and live in the same city have the same name."

2) Transitivity

If $\texttt{X} \rightarrow \texttt{Y}$ and $\texttt{Y} \rightarrow \texttt{Z}$, then $\texttt{X} \rightarrow \texttt{Z}$

Example :

$\texttt{R(SIN, ZIPCODE, CITY)}$

$\{\texttt{SIN} \rightarrow \texttt{ZIPCODE, ZIPCODE} \rightarrow \texttt{CITY}\} \models \texttt{SIN} \rightarrow \texttt{CITY}$

"If we know the zipcode from a sin number and the city form the zipcode then we know the city from the sin number."

3) Augmentation

$\texttt{X} \rightarrow \texttt{Y} \models \texttt{XZ} \rightarrow \texttt{YZ}$

Example:

$\texttt{SIN} \rightarrow \texttt{ZIPCODE} \models \texttt{SIN CITY} \rightarrow \texttt{ZIPCODE CITY}$

4) Union et decomposition

$$\{ \mathsf{X} \rightarrow \mathsf{A}, \mathsf{X} \rightarrow \mathsf{B} \} \Leftrightarrow \mathsf{X} \rightarrow \mathsf{AB}$$

5) Pseudotransitivity

$$\{ \mathsf{X} \rightarrow \mathsf{Y}, \mathsf{WY} \rightarrow \mathsf{Z} \} \models \mathsf{WX} \rightarrow \mathsf{Z}$$

REMARK: Union, decomposition and pseudotransitivity can be deduced from the others axioms.

Closing set of a functional dependencies set

From the functional dependencies set given from the real world and using the preceding properties (`Armstrong's axioms`) we can deduce others FDs:

Examples:

1) R(A,B,C,D)

$\mathcal{F} = \{ A \rightarrow B,\ B \rightarrow C \}$

Transitivity: $A \rightarrow C$

Notation: $\mathcal{F} \models A \rightarrow C$

Functional Dependencies

2) R(Course, Prof, Hour, Room, Student, Mark)

$\mathcal{F}$ = { C $\rightarrow$ P, HR $\rightarrow$ C, HP $\rightarrow$ R, CS $\rightarrow$ M, HS $\rightarrow$ R }

$\mathcal{F} \models$ HR $\rightarrow$ P, HS $\rightarrow$ C,P,M

So HS is a **key** Why?

The union of $\mathcal{F}$ and every deduced FD is called **closing set**, or **attribute closure** of $\mathcal{F}$ is denoted $\mathcal{F}^+$.

Minimal Closure

The minimal closure $\mathcal{G}$ of a set $\mathcal{F}$ of functional dependencies (FD) is a set of FD such that:

1. We can infer from $\mathcal{G}$ the same FD than from $\mathcal{F}$: $\mathcal{G}^+ = \mathcal{F}^+$
2. Only one attribute is on the right side on every FD in $\mathcal{G}$ (decomposition)
3. Every FD are useful: if we delete one we can not obtain $\mathcal{F}^+$ anymore.
4. Every FD are elemantary: the set $\{\mathtt{A} \rightarrow \mathtt{C};\ \mathtt{AC} \rightarrow \mathtt{B}\}$ is redundant: we replace $\mathtt{AC} \rightarrow \mathtt{B}$, which is not elementary by $\mathtt{A} \rightarrow \mathtt{B}$.

Example

Functional Dependencies	**Minimal Closure**
$AB \rightarrow C$	$AB \rightarrow C$
$C \rightarrow A$	$C \rightarrow A$
$BC \rightarrow D$	$BC \rightarrow D$
$ACD \rightarrow B$	$CD \rightarrow B$
$D \rightarrow EG$	$D \rightarrow E$
$BE \rightarrow C$	$D \rightarrow G$
$CE \rightarrow AG$	$BE \rightarrow C$
	$CE \rightarrow G$

$CE \rightarrow A$ is missing, why?

$ACD \rightarrow B$ is not elementary, why?

Answer:

$\mathsf{ACD} \rightarrow \mathsf{B}$ is not elemenaty, why?

$\mathsf{C} \rightarrow \mathsf{A} \models \mathsf{CCD} \rightarrow \mathsf{ACD}$ by *augmentation*

$\mathsf{CCD} \rightarrow \mathsf{ACD} \models \mathsf{CD}\rightarrow \mathsf{ACD}$ ($\mathsf{A} \rightarrow \mathsf{A}^i$ by reflexivity and union)

$\mathsf{CD} \rightarrow \mathsf{ACD}$ and $\mathsf{ACD} \rightarrow \mathsf{B} \models \mathsf{CD} \rightarrow \mathsf{B}$ by *transitivity*

Second demonstration

By pseudotransitivity, $\{\ \mathtt{C} \rightarrow \mathtt{A},\ \mathtt{ACD} \rightarrow \mathtt{B}\} \models \mathtt{CCD} \rightarrow \mathtt{B} \models \mathtt{CD} \rightarrow \mathtt{B}$.

Functional Dependencies

Minimal Closure

Concretely, the minimal closure

- is either directly given by the real world analysis.
- is eitheir trivialy deduced because:
  - every FD are useful and elementary
  - It is enough to decompose the right side of every FDs.

New Definition for a SuperKey

Let $\mathsf{R(U)}$ a relational schema, $\mathcal{F}$ a set of functional dependencies and $\mathsf{X} \subseteq \mathsf{U}$ a set of attributes. X is a superkey of R if for all $\mathsf{A} \in \mathsf{U}$

$$\mathcal{F} \models \mathsf{X} \rightarrow \mathsf{A}$$

or if:

$$\mathsf{X} \rightarrow \mathsf{A} \in \mathcal{F}^{+}$$

Compute $\mathcal{F}^{+}$ from $\mathcal{F}$ might take time.

But show that $\mathcal{F} \models \mathsf{X} \rightarrow \mathsf{A}$ is easy and fast.

Calculation of a key of a relation

Let $\mathsf{R(U)}$ a relational schema and $\mathsf{X} \subseteq \mathsf{U}$ a set of attributes.

- We define X^+ as a subset of attributes A such that $\mathcal{F} \models \mathsf{X} \rightarrow \mathsf{A}$
- If A belongs to X^+, then by definition $\mathcal{F} \models \mathsf{X} \rightarrow \mathsf{A}$
- X^+ is the set of functionally dependent attributes of X.

To find a key we use the following algorithm:

1. We search for one X such that $X^+ = \mathsf{U} \Rightarrow \mathsf{X}$ is a superkey
2. X is a key, if it does not exist $\mathsf{Y} \subset \mathsf{X}$ such that $Y^+ = \mathsf{U}$

$\boxed{X^+}$

Step 1: We start from X

For every Y $\rightarrow$ A, such that Y $\subseteq$ X, Y $\rightarrow$ A $\in \mathcal{F}$, we add A to X.

We get X^1.

Step i: We start X^{i-1}

For every Y $\rightarrow$ A, such that Y $\subseteq$ X^{i-1}, Y $\rightarrow$ A $\in \mathcal{F}$, we add A to X^{i-1}.

We get X^i.

We stop when we do not find new FDs anymore.

$$X^+ = X^{i+1} = X^i$$

Examples

1. Show that `HS` is a key for `R(CHSNRP)` with the set $\mathcal{F}$ of FD given before.
2. for the relation `ADDRESS(CITY, STREET, NUMBER, ZIPCODE)`, show that `CITY STREET NUMBER` is a key. What is the other one?

UPDATE ANOMALIES

Example

Let the schema S1:

```
Supplier(SNAME, FADDRESS)
Product(SNAME, PNAME, PRICE)
```

and the set of DF: $\mathcal{F} = \{\texttt{SNAME} \rightarrow \texttt{FADDRESS}, \texttt{(SNAME PNAME)} \rightarrow \texttt{PRICE}\}$

Suppose now that we replace S1 by S2:

```
R(SNAME, SADDRESS, PNAME, PRICE)
```

Anomalies

R(SNAME, SADDRESS, PNAME, PRICE)

$\mathcal{F} = \{\texttt{SNAME} \rightarrow \texttt{SADDRESS}, (\texttt{SNAME PNAME}) \rightarrow \texttt{PRICE}\}$

What is the key of R?

SMITH	Toronto	COMPUTER	1000
JONS	Montreal	COMPUTER	900
SMITH	Toronto	KEYBOARD	400

1) REDONDANCY: the address for one supplier appears several times.

2) UPDATE: if we change the address in one tuple, we must also perform the same update in the others.

3) DELETION: if `JONS` do not supply `COMPUTER` anymore, we delete the second tuple we lost any information about JONS.

4) INSERTION: we can not insert a new supplier and its address if we do not know, at least, one product that it supplies.

SMITH	Toronto	COMPUTER	1000
JONS	Montreal	COMPUTER	900
SMITH	Toronto	KEYBOARD	400
DURAND	NICE		

$\Longrightarrow$ THE INITIAL SCHEMA S1 IS “BETTER”

Integrity Constraints

The list of attributes is not sufficient to describe the semantic of the real world.

It exists several types of constraints on the tuples:

1. dependencies (functional, multivalued, ...)
2. constraints that depend of the attribute domain: year < 2000
3. etc.

It is the dependencies which permits a *good schema conception*, i.e., the decomposition in “good” relations.

Qualities for a good schema

1. Avoid anomalies $\Longrightarrow$ decomposition
2. The decomposition should keep the same amount of information
   Join a relation
   `f1` *of the schema* `SUPPLIER(SNAME, SADDRESS)`
   and a relation
   `f2` *of the schema* `SUPPLIER(SNAME, PNAME, PRICE)`
   got by decomposition of the relation
   `r` *of the schema* `R(SNAME, SADDRESS, PNAME, PRICE)`
   must give back `r`.
3. The decomposition must keep the same constraints (FD). The decomposition of `R` in `R1(SNAME, SADDRESS, PRICE)` and `R2(PNAME, PRICE)` do not preserve the FD. Why?

DECOMPOSITION AND NORMAL FORM

Relation in first normal form (1NF)

- All attributes are atomic (*elementary*)
- Relation that we know.

Relation 1NF

```
MARKS ( COURSE    STUDENT    MARK )
        DB        John        80
        DB        Mark        90
        DB        Tom          0
        ARCHI     Tom        100
        ARCHI     John         0
```

Relation N1FN

```
MARKS ( COURS     PERF (STUDENT  MARK) )
        DB              John        80
                        Mark        90
                        Tom          0
        ARCHI           Tom        100
                        John         0
```

Relation in 3rd Normal Form

- **2nd Normal Form**:
  Purely historical.
- **3rd normal form: 3NF**
  - avoid most of the anomalies

  Goal of the game: to decompose a relation (1NF) to a set of 3NF relations.

3NF: First Definition

Let `(R, F)` be a relational schema.

We suppose that $\mathcal{F}$ is a minimal closure.

Definition: R is in 3NF, if for every $\mathsf{X} \rightarrow \mathsf{A}$ of $\mathcal{F}$,

- either X is a key
- or A belongs to one of the keys.

Examples: 3rd Normal Form

1) Post(City, Street, Zipcode)

$\mathcal{F} = \{ \texttt{CS} \rightarrow \texttt{Z}, \ \texttt{Z} \rightarrow \texttt{C} \}$

Keys: `CS, SZ`

`Post` is in 3NF.

2) Supplier(SNAME, ADDR, PNAME, PRICE)

$\mathcal{F} = \{ \texttt{SNAME} \rightarrow \texttt{ADDR}, \ \texttt{SNAME PNAME} \rightarrow \texttt{PRICE} \}$

Keys: `(SNAME PNAME)`

`Supplier` is not in 3NF.

3) Schedule(Course, Hour, Room)

$\mathcal{F} = \{ \texttt{RH} \rightarrow \texttt{C}, \ \texttt{C} \rightarrow \texttt{R} \}$

Keys: `RH, CH`

`Schedule` is in 3NF.

4) R(A, B, C, D)

$\mathcal{F} = \{ \texttt{AB} \rightarrow \texttt{C}, \ \texttt{B} \rightarrow \texttt{D}, \ \texttt{BC} \rightarrow \texttt{A} \}$

Keys: `AB, BC`

`R` is not in 3NF.

3NF: Second Definition

Remark: It is not necessary for $\mathcal{F}$ to be a minimal closure. It is enough that for all FD $\mathsf{X} \rightarrow \mathsf{A}$ of $\mathcal{F}^+$,

- A is only composed by one attribute.
- A is not one of the attribute of X

The definition of a 3NF schema becames:

Definition: A relation R is in 3NF, if every FD $\mathsf{X} \rightarrow \mathsf{A}$ of $\mathcal{F}^+$ satisfies the preceding conditions,

- either X is a *super*key
- or A belongs to one of the keys.

Actually, it is not necessary to check *all* FD of $\mathcal{F}^+$.

It is enough to check the ones belonging to $\mathcal{F}$!

RemarK (Cont'd):

Why *super*key in the first condition?

With conditions 1 and 2 weaker than the condition on the minimal closure it may exist some non elementary conditions:

If $\mathsf{X} \rightarrow \mathsf{A}$ is a non elementary condition
and if $\mathsf{X}^+ = \mathsf{U}$, then X is a *super*key.

Remark (end):

R(A,B,C,D)

$\mathcal{F}$ = { AB $\rightarrow$ C, B $\rightarrow$ D, D $\rightarrow$ B, B $\rightarrow$ A }

$\mathcal{F}$ is not a minimal closure, why?

(R, $\mathcal{F}$) is in 3NF, why?

Lossless-Join Decomposition

Example

R	A	B	C
	a	b	c
	a	b	a
	c	b	d

and $\mathcal{F} = \{ \mathtt{A} \rightarrow \mathtt{B} \}$

We decompose in:

R1	A	B
	a	b
	c	b

R2	B	C
	b	c
	b	a
	b	d

R1 = $\pi_{A,B}$(R)
R2 = $\pi_{B,C}$(R)
R′ = R1 $\bowtie$ R2 $\neq$ R:

R′	A	B	C
	a	b	c
	a	b	a
	a	b	d
	c	b	c
	c	b	a
	c	b	d

The decomposition of R in R1 and R2 losses information.

The join creates tuples that do not exist in R.

Now we decompose $\mathtt{R}'$ in:

R1′(	A	B)
	a	b
	c	b

R2′ (	A	C)
	a	c
	a	a
	c	d

$\mathtt{R}'' = \mathtt{R1}' \bowtie \mathtt{R2}' = \mathtt{R}'$:

R′′ (	A	B	C)
	a	b	c
	a	b	a
	c	b	d

this decomposition is *lossless-join*.

The condition is that after the join we found the same information than before the decomposition.

Definition A decomposition of R in R1, R2, ..., Rk with regard to a set of FD $\mathcal{F}$ is lossless-join iff for every instance r of the schema R that *satisfies*$\mathcal{F}$, we have:

$$r = \pi_{R1}(r) \bowtie \pi_{R2}(r) \ldots \bowtie \pi_{Rk}(r)$$

Theorem:

If (R1, R2) is decomposition of R and $\mathcal{F}$ a set of FD, then (R1, R2) is lossless-join w.r.t. $\mathcal{F}$, iff:

$$R1 \cap R2 \rightarrow R1 - R2$$

or

$$R1 \cap R2 \rightarrow R2 - R1$$

belongs to $\mathcal{F}^+$.

Examples

$$R\ (A,\ B,\ C)$$

$$\mathcal{F} = \{ A \rightarrow B \}$$

1) R1(A, B), R2(B, C)

$AB \cap BC = B$

$AB - BC = A$

$BC - AB = C$

The FD $B \rightarrow A$ does not exist, nor $B \rightarrow C$ in $\mathcal{F}^+$

$\Rightarrow$ **The decomposition loses information.**

2) R1(A, B), R3(A, C)

$AB \cap AC = A$

$AB - AC = B$

$A \rightarrow B$ is $\mathcal{F}$ ($\mathcal{F}^+$).

$\Rightarrow$ **the decomposition is lossless join.**

Dependency Preserving Decomposition

Definitions

1. Projection of a set of FD om $\mathtt{Z} \subset \mathtt{U}$
   $\pi_Z(\mathcal{F}) = \{X \rightarrow Y \in \mathcal{F}^+ \mid XY \subseteq Z\}$
   Example: $\mathtt{R(A,B,C,D)}$, $\mathcal{F} = \{ \mathtt{AB} \rightarrow \mathtt{C}, \mathtt{C} \rightarrow \mathtt{A}, \mathtt{A} \rightarrow \mathtt{D} \}$
   $\pi_{ABC}(\mathcal{F}) = \{ \mathtt{AB} \rightarrow \mathtt{C}, \mathtt{C} \rightarrow \mathtt{A} \}$
2. Decomposition which preserves the FDs of $\mathcal{F}$
   Let $\Delta = \mathtt{(R1,...,Rk)}$ be a decomposition, and $\mathcal{F}$ a set of FD.
   Δ preserves the FDs of $\mathcal{F}$, if we can find again every FDs of $\mathcal{F}^+$ from the union $\mathcal{G}$ of all FDs projected from $\mathcal{F}$ in $\pi_{R1}(\mathcal{F}), \ldots, \pi_{Rk}(\mathcal{F})$:
   $\mathcal{G}^+ = \mathcal{F}^+$

Examples

R(A,B,C,D)

$\mathcal{F}$ = { AB $\rightarrow$ C, C $\rightarrow$ A, A $\rightarrow$ D }

Δ = (ABC, BD) do not preserve the FDs of $\mathcal{F}$

Δ = (ABC, AD) preserves the FDs of $\mathcal{F}$

R(A,B,C)

$\mathcal{F}$ = { A $\rightarrow$ B, B $\rightarrow$ A, A $\rightarrow$ C }

Δ = (AB, BC) preserve the FDs of $\mathcal{F}$

R(A, B, C, D)

$\mathcal{F}$ = { $\mathtt{A} \rightarrow \mathtt{B}$, $\mathtt{B} \rightarrow \mathtt{C}$, $\mathtt{AB} \rightarrow \mathtt{D}$ }

The decomposition

```
R1(AC)
R2(AB)
R3(CD)
```

do not preserve the FDs of $\mathcal{F}$. Why?

Decomposition of a relation in 3NF

Given a schema `(R, F)` not in 3NF, i.e. with some anomalies, we want to find a decomposition of `R`:

1. with 3NF relations;
2. lossless-join;
3. preserve the FDs of $\mathcal{F}$

Remark:

- a lossless-join decomposition do not necessarily preserve the FDs and inversely.
- the result does not necessarily give relations in 3NF.

Theorem: Every 1NF relation has a decomposition in 3NF relations which are lossless-join and preserve the functional dependencies.

Algorithm

We assume that $\mathcal{F}$ is a minimal closure.

1. For each $\texttt{X} \rightarrow \texttt{A} \in \mathcal{F}$, create a relation of schema `(XA)`.
2. If no keys is contained in one a the schema created during the first step, add a relation of schema `Y` where `Y` is one key.
3. If after the first step, it exist one relation `R1` with a schema `(X1A1)` contained in a schema `(X2A2)` of another relation `R2`, delete the relation `R1`.
4. Replace the relations $(\texttt{XA}_1), \ldots, (\texttt{XA}_k)$ (corresponding to FD with the same left member) by a unique relation: $(\texttt{XA}_1 \ldots \texttt{A}_k)$.

Examples

1) R(A,B,C,D)

$\mathcal{F}$ = { AB → C, B → D, C → A }

Keys: AB, BC

- Step 1: R1(ABC) R2(BD) R3(CA)
- Step 2: No need to create a new relation:

  the key AB belongs to R1

- Step 3: Delete R3: CA ⊂ ABC

Good decomposition: R1(ABC) R2(BD)

We can check that R1 et R2 are in 3NF.

2) R(A,B,C,D,E)

$\mathcal{F} = \{ \texttt{AB} \rightarrow \texttt{C}, \texttt{C} \rightarrow \texttt{D}, \texttt{C} \rightarrow \texttt{A} \}$ **Keys**: `ABE, BCE`

- Step 1: `R1(ABC)` `R2(CD)` `R3(CA)`
- Step 2: We add a relation of schema for the key `ABE`: `R4(ABE)`
- Step 3: Delete `R3`: $\texttt{CA} \subset \texttt{ABC}$

Good decomposition: `R1(ABC)` `R2(CD)` `R4(ABE)`

Other solutions:

- Step 4: We replace R2 and R3 from step 1 by the relation of schema `(CAD)`

Other good decomposition: `R1(ABC)` `R2'(CAD)` `R4(ABE)`

What happened if we have chosen the key CBE?

3) R(A,B,C,D)

$\mathcal{F} = \{ \mathtt{AB} \rightarrow \mathtt{C}, \mathtt{C} \rightarrow \mathtt{D}, \mathtt{C} \rightarrow \mathtt{A}, \mathtt{AB} \rightarrow \mathtt{D} \}$

Keys: `BA, BC`

The relation is not in 3NF. Why?

- Step 1: `R1(ABC)` `R2(CD)` `R3(CA)` `R4(ABD)`
- Step 2: we do not add the relation: key $\mathtt{AB} \subseteq$ `R1(ABC)`
- Step 3: Delete `R3`: $\mathtt{CA} \subseteq \mathtt{ABC}$
- Step 4: We replace `R1` et `R4` by `R5(ABCD)` $\Rightarrow$ we can delete `R2`.

Decomposition: `R5(ABCD)`

This decomposition is not in 3NF. Where is the problem?

Boyce-Codd Normal Form (BCNF)

Some anomalies still exist in 3NF.

Example: `Post(City,Street,ZipCode)`, $\mathcal{F} = \{\, \texttt{VC} \rightarrow \texttt{Z}, \texttt{V} \rightarrow \texttt{C} \,\}$

Keys: `CS, SZ`

```
Post  ( City        Street    ZipCode  )
        Toronto     Queen     M4F3G4
        Toronto     King      M4F3G4
```

$\Rightarrow$ Redondancy between the zipcode and the city.

Definition: A relation is in *Boyce-Codd* Normal Form (BCNF), if for every functional dependency of $\mathcal{F}$, the left member id a superkey.

Interest: We eliminate all anomalies.

Remark: Every BCNF relation is in 3NF.

Unfortunately, it does not always exist a decomposition in BCNF which also:

- is lossless-join
- preserves the FD.

Post Example

Post(City,Street,ZipCode), $\mathcal{F} = \{ \mathtt{CS} \rightarrow \mathtt{Z}, \mathtt{Z} \rightarrow \mathtt{C} \}$

Keys: CS, SZ

R is in 3NF but not in BCNF (in $\mathtt{Z} \rightarrow \mathtt{C}$, Z is not a key)

```
Post  ( City        Street    ZipCode  )
        Toronto     Queen     M4F3G4
        Toronto     King      M4F3G4
```

The decomposition `R1(City,ZipCode), R2(Street,ZipCode)` avoids the redondancy `City,ZipCode`, it is lossless-join, but does not preserve the functional dependency $\mathtt{CS} \rightarrow \mathtt{Z}$

```
R1 ( City       ZipCode  )     R2 ( Strret        Code  )
     Toronto    M                   Queen         M
     Montreal   T                   Queen         T
```

The insertion of `Toronto Queen M4`, i.e. `Toronto M4` and `King M4` respects $\mathtt{Z} \rightarrow \mathtt{C}$ but do not respect anymore $\mathtt{CS} \rightarrow \mathtt{Z}$

```
R1 ( Ville      Code  )        R2 ( Street        Code  )
     Toronto    M                   Queen         M
     Montreal   T                   Queen         T
     Toronto    M4                  Queen         M4
```

Decomposition Algorithm

We assume that R(U) is a relational schema and $\mathcal{F}$ is a minimal cover.

1. Pick a FD X $\rightarrow$ Y not verifying BCNF
2. Partition R into R1(X Y) and R2(X (U-Y))
3. If R1 is not in BCNF start the algorithm with R1 in input
4. If R2 is not in BCNF start the algorithm with R2 in input

Example 1

$\mathtt{R(BOSQID)}$, $\mathcal{F} = \{\, \mathtt{IS} \rightarrow \mathtt{Q}, \mathtt{B} \rightarrow \mathtt{O}, \mathtt{I} \rightarrow \mathtt{B}, \mathtt{S} \rightarrow \mathtt{D} \,\}$ **Candidate Key:** IS

Pick $\mathtt{B} \rightarrow \mathtt{O}$:

- R1 (BO)
- R2 (BSQID)

Now we decompose R2 using $\mathtt{S} \rightarrow \mathtt{D}$

- R3 (SD)
- R4 (BSQI)

Now we decompose R4 using $\mathtt{I} \rightarrow \mathtt{B}$

- R5(IB)
- R6(ISQ)

Set of relational schema which are in BCNF from R and $\mathcal{F}$:

- R1 (BO) $\mathcal{F}_1 = \{ \texttt{B} \rightarrow \texttt{O} \}$
- R3 (SD) $\mathcal{F}_3 = \{ \texttt{S} \rightarrow \texttt{D} \}$
- R5 (IB) $\mathcal{F}_5 = \{ \texttt{I} \rightarrow \texttt{B} \}$
- R6 (ISQ) $\mathcal{F}_6 = \{ \texttt{IS} \rightarrow \texttt{Q} \}$

Example 2

$\mathtt{R(ABCDEF)}$, $\mathcal{F} = \{\ \mathsf{A} \rightarrow \mathsf{BC}, \mathsf{D} \rightarrow \mathsf{AF}\ \}$ **Candidate Key:** DE

- R1(ABC) $\mathcal{F}_1 = \{\ \mathsf{A} \rightarrow \mathsf{BC}\ \}$
- R2(ADF) $\mathcal{F}_2 = \{\ \mathsf{D} \rightarrow \mathsf{AF}\ \}$
- R3(DE) $\mathcal{F}_3 = \emptyset$

Example 3

R(ABC), $\mathcal{F}$ = { AB $\rightarrow$ C, C $\rightarrow$ A } **Candidate Key:** AB,CB

- R1(AC) $\mathcal{F}_1$ = { C $\rightarrow$ A }
- R@(DE) $\mathcal{F}_2 = \emptyset$

4NF

Definition

- Functional dependencies rule out certain tuples from appearing in a relation: if A $\rightarrow$ B, then we cannot have two tuples with the same A value but different B values.
- Multivalued dependencies do not rule out of the existence of certain tuples. Instead they require that other tuples of a certain form be present in the relation.
- Let R(U, $\mathcal{F}$) be a relational schema and A,B $\subseteq$ U. The multivalued dependency:

$$A \twoheadrightarrow B$$

holds in R if any legal relation r(R), for all pairs of tuples t_1 and t_2 in r such that t_1**[A]** = t_2**[A]**, there exist tuples t_3 and t_4 in r such that:

  - t_1**[A]** = t_2**[A]** = t_3**[A]** = t_4**[A]**
  - t_1**[B]** = t_3**[B]**
  - t_2**[R-B]** = t_3**[R-B]**
  - t_2**[B]** = t_4**[B]**
  - t_4**[R-B]** = t_1**[R-B]**

Example

name	address	car
Tom	North Rd.	Toyota
Tom	Oak St.	Honda
Tom	North Rd.	Honda
Tom	Oak St.	Toyota

$$MVD = \{name \twoheadrightarrow address, name \twoheadrightarrow car\}$$

Example

name	street	city	title	year
C. Fisher	123 Maple St.	Hollywood	Star Wars	1977
C. Fisher	5 Locust Ln.	Malibu	Star Wars	1977
C. Fisher	123 Maple St.	Hollywood	Empire Strike Back	1980
C. Fisher	5 Locust Ln.	Malibu	Empire Strike Back	1980
C. Fisher	123 Maple St.	Hollywood	Return of the Jedi	1983
C. Fisher	5 Locust Ln.	Malibu	Return of the Jedi	1983

$$\mathcal{F} = \{name\ street\ title\ year \rightarrow city\}$$
$$MVD = \{name \twoheadrightarrow street\ city\}$$

Theory of Multivalued Dependencies

We will need to compute all the multivalued dependencies that are logically implied by a given set of multivalued dependencies.

- Let D denote a set of functional and multivalued dependencies.
- The closure of D^+ is the set of all functional and multivalued dependencies logically implied by D.
- We can compute from D^+ using the formal definitions, but it is easier to use a set of inference rules.

Theory of Multivalued Dependencies

The following set of inference rules is sound and complete. The first three rules are Armstrong's axioms.

- Reflexivity rule: if X is a set of attributes and Y $\subseteq$ X, then $X \to Y$ holds.
- Augmentation rule: $X \to Y \models XZ \to YZ$.
- Transitivity rule: if $X \to Y$ holds, and $Y \to Z$ holds, then $X \to Z$ holds.
- Complementation rule: if $X \twoheadrightarrow Y$ holds, then $X \twoheadrightarrow (R - B - A)$ holds.
- Multivalued augmentation rule: if $X \twoheadrightarrow Y$ holds, and $Z \subseteq R$ and $T \subseteq Z$, then $XZ \twoheadrightarrow TB$ holds.
- Multivalued transitivity rule: if $X \twoheadrightarrow Y$ holds, and $Y \twoheadrightarrow Z$ holds, then $X \twoheadrightarrow Z - Y$ holds.
- Replication rule: if $X \to Y$ holds, then $X \twoheadrightarrow Y$.
- Coalescence rule: if $X \to Y$ holds, and $Z \subseteq Y$, and there is a T such that$T \subseteq R$ and $T \cap Y = \emptyset$ and $T \to Z$, then $A \to Z$ holds.

An example of *multivalued transitivity rule* is as follows. If we have R(A,B,C,D) and $A \twoheadrightarrow B$ and $B \twoheadrightarrow B, C$. Thus we have $A \twoheadrightarrow C$, where $C = B, C - B$

An example of *coalescance rule* is as follows. If we have R(A,B,C,D) and $A \twoheadrightarrow B, C)$ and $D \rightarrow B$, then we have $A \rightarrow B$

A MVD $X_1X_2...X_n \twoheadrightarrow Y_1Y_2...Y_m$ for a relation R is *nontrivial* if:

1. None of the Y's is among the A's.
2. Not all the attributes of R are among the A's and B's.

Other Axioms

- Multivalued Union rule: if $X \twoheadrightarrow Y$ holds and $X \twoheadrightarrow Z$ holds, then $X \twoheadrightarrow Y, Z$ holds.
- Intersection rule: If $X \twoheadrightarrow Y$ holds and $X \twoheadrightarrow Z$ holds, then $X \twoheadrightarrow Y \cap Z$ holds.
- Difference rule: If $X \twoheadrightarrow Y$ holds, and $X \twoheadrightarrow Z$, then $X \twoheadrightarrow Y - Z$ holds and $X \twoheadrightarrow Z - Y$ holds.

Fourth Normal Form (4NF)

- We saw that a BCNF schema was not an ideal design as it suffered from repetition of information.
- We can use the given multivalued dependencies to improve the database design by decomposing it into **fourth normal form**.
- A relation schema R is in 4NF with respect to a set D of functional and multivalued dependencies if for all multivalued dependencies in D^+ of the form $X \twoheadrightarrow Y$, where $X \subseteq R$ and $Y \subseteq R$:
  - $X \twoheadrightarrow Y$ is a non trivial multivalued dependency; and
  - X is a superkey for schema R.
- A database design is in 4NF if each member of the set of relation schemas is in 4NF.
- The definition of 4NF differs from the BCNF definition only in the use of multivalued dependencies:
  - Every 4NF schema is also in BCNF.

Summarize

Property	3NF	BCNF	4NF
Eliminates redundancy due to FD's	Most	Yes	Yes
Eliminates redundancy due to MVD's	No	No	Yes
hline Preserves FD's	Yes	Maybe	Maybe
Preserves MVD's	Maybe	Maybe	Maybe