Data Bases Data Mining

Foundations of databases: from functional dependencies to normal forms

Database Group

Dpt Informatique

LIRIS

CNRS

http://liris.cnrs.fr/ecoquery/dokuwiki/doku.php?id=enseignement:dbdm:start

March 1, 2017

Exemple

Let $\mathcal{U} = \{id, name, address, cnum, desc, grade\}$ a set of attributes to model students and courses. We consider the following database schemas :

- $R1 = \{Data\}$ with $schema(Data) = \mathcal{U}$[1].
- $R2 = \{Student, Course, Enrollment\}$ avec
  - $schema(Student) = \{id, name, address\}$
  - $schema(Course) = \{cnum, desc\}$
  - $schema(Enrollment) = \{id, cnum, grade\}$

How to compare these schemas?

- Which one is the "best"?
- Why?

[1] Similar to a spreadsheet.

Exemple

Data	*id*	*name*	*address*	*cnum*	*desc*	*grade*
	124	Jean	Paris	F234	Philo I	A
	456	Emma	Lyon	F234	Philo I	B
	789	Paul	Marseille	M321	Analyse I	C
	124	Jean	Paris	M321	Analyse I	A
	789	Paul	Marseille	CS24	BD I	B

Is there any problem here?

Exemple

Data	*id*	*name*	*address*	*cnum*	*desc*	*grade*
	124	Jean	Paris	F234	Philo I	A
	456	Emma	Lyon	F234	Philo I	B
	789	Paul	Marseille	M321	Analyse I	C
	124	Jean	Paris	M321	Analyse I	A
	789	Paul	Marseille	CS24	BD I	B

Is there any problem here?

Redundancies!

Redundancies

Data	*id*	*name*	*address*	*cnum*	*desc*	*grade*
	124	Jean	Paris	F234	Philo I	A
	456	Emma	Lyon	F234	Philo I	B
	789	Paul	Marseille	M321	Analyse I	C
	124	Jean	Paris	M321	Analyse I	A
	789	Paul	Marseille	CS24	BD I	B

Intuition on functional dependencies

- A student' *id* gives her/his name and address, so for each new enrollment, his/her name and address are duplicated!
- $\pi_{id,name,address}(Data)$ is the graph of a (partial) function $f : id \rightarrow name \times address$, similarly for $\pi_{cnum,desc}(Data)$
- $R2 = \{Student, Course, Enrollment\}$ is better than $R1 = \{Data\}$ because it avoids redundancies by keeping unrelated information (e.g., a student's name and a course' description) unrelated...

Functional is a theoretical tool to capture and reason on this phenomenon.

Functional dependencies: definition

Syntax

A *Functional Dependency (FD)* over a relation schema R is a formal expression of the form[2], with $X, Y \subseteq R$:

$$R : X \rightarrow Y$$

- $X \rightarrow Y$ is read "X functionally determines Y" or "X gives Y"
- A FD $X \rightarrow Y$ is trivial when $Y \subseteq X$
- A FD is standard when $X \neq \emptyset$.
- A set of attributes X is a key when $R : X \rightarrow R$

Semantics

Let r be a relation (a.k.a. *instance*) over R. The FD $R : X \rightarrow Y$ is *satisfied* by r, written $r \models R : X \rightarrow Y$, *iff*

$$\forall t_1, t_2 \in r. t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$$

[2]We write $X \rightarrow Y$ when R is clear from the context.

What constraint is implied by a *non-standard* FD?

What constraint is implied by a *non-standard* FD?

Why a *trivial* FD is said to be *trivial*?

Example

r	**A**	**B**	**C**	**D**
t_1	a_1	b_1	c_1	d_1
t_2	a_1	b_1	c_1	d_2
t_3	a_1	b_2	c_2	d_3
t_4	a_2	b_2	c_3	d_4

- $r \models AB \to C$ (no counter-example)
- $r \models D \to ABCD$ (no counter-example)
- $r \not\models AB \to D$ (e.g., $t_1[AB] = t_2[AB]$ but $t_1[D] \neq t_2[D]$)
- $r \not\models A \to C$ (e.g., $t_2[A] = t_3[A]$ but $t_2[C] \neq t_3[C]$)

Checking if a FD $R : X \to A$ holds in an instance

Using SQL (of course), with $X = \{A_1, \ldots, A_n\}$

```
SELECT A1, ..., An COUNT(DISTINCT A) AS NB
FROM R
GROUP BY A1, ..., An
HAVING COUNT(DISTINCT A) > 1;
```

Logical implication

Definition

Let F be a set of FDs on a relation schema R and let f be a single FD on R. We overload $\models$ for a set of FDs:

$$r \models F \text{ iff } \forall f \in F . r \models f$$

F logical (semantically) implies f, written

$$F \models f \text{ iff } \forall r . r \models F \Rightarrow r \models f$$

Example

With $F = \{A \to BCD, BC \to E\}$ and $r \models F$, the following hold as well:

- $r \models A \to CD$
- $r \models A \to E$

It can be proved using the definition of $\models$ and basic reasoning on projection of tuples.

Armstrong's System for FD

Armstrong's System

The following rules constitute the so call *Armstrong's system* for FDs:

- Reflexivity

$$\frac{Y \subseteq X}{X \to Y}$$

- Augmentation

$$\frac{X \to Y}{WX \to WY}$$

- Transitivity

$$\frac{X \to Y \qquad Y \to Z}{X \to Z}$$

Proof using Armstrong's system

Example

Let $\Sigma = \{A \to B, B \to C, CD \to E\}$ be a set of FDs on $\{A, B, C, D, E\}$.
We show that $\Sigma \vdash AD \to E$

$$\dfrac{\dfrac{\dfrac{A \to B \qquad B \to C}{A \to C}}{AD \to CD} \qquad CD \to E}{AD \to E}$$

Properties

Soundness and completeness

- The system is sound if $F \vdash f \Rightarrow F \models f$
  if there is a proof, the proof is valid
- The system is complete if $F \models f \Rightarrow F \vdash f$
  if it's valid, there is a proof

$$F \models \alpha \Leftrightarrow F \vdash \alpha$$

Soundness

Prove for every rule that, if its hypothesis are valid then its conclusion is valid as well.

Example: transitivity

Let r be ans instance on R s.t. $r \models X \rightarrow Y$ et $r \models Y \rightarrow Z$. Let $t_1, t_2 \in r$ be two tuples in r s.t. $t_1[X] = t_2[X]$, we have to show that $t_1[Z] = t_2[Z]$. Using $r \models X \rightarrow Y$ we deduce that $t_1[Y] = t_2[Y]$, then using $r \models Y \rightarrow Z$ we deduce that $t_1[Z] = t_2[Z]$. So the transitivity of FDs amounts to the transitivity of equality...

Additional rules

- Decomposition

$$\frac{X \rightarrow YZ}{X \rightarrow Y}$$

- Composition

$$\frac{X \rightarrow Y \qquad X \rightarrow Z}{X \rightarrow YZ}$$

- Pseudo-transitivity

$$\frac{X \rightarrow Y \qquad WY \rightarrow Z}{WX \rightarrow Z}$$

This rules are sound and can be (safely) added to Armstrong's system

Completeness

Formal proofs

A (formal) proof of f from Σ using Armstrong' system written $\Sigma \vdash f$ is a *sequence* $\langle f_0, \ldots, f_n \rangle$ of FDs s.t. $f_n = f$ et $\forall i \in [0..n]$:

- either $f_i \in \Sigma$;
- or f_i is the *conclusion* of a rule of which all its *antecedents* $f_0 \ldots f_p$ appear before f_i in the sequence.

Completeness: $\Sigma \models X \to Y \Rightarrow \Sigma \vdash X \to Y$

We need a clear distinction between

- the semantic closure of X: $X^+ = \{A \mid \Sigma \models X \to A\}$
- the syntactic closure of X: $X^\star = \{A \mid \Sigma \vdash X \to A\}$

Lemma: $\Sigma \vdash X \to Y \Leftrightarrow Y \subseteq X^\star$

Completeness

$$
\begin{aligned}
& \Sigma \models X \to Y \Rightarrow \Sigma \vdash X \to Y \\
\equiv\ & \Sigma \nvdash X \to Y \Rightarrow \Sigma \not\models X \to Y \\
\equiv\ & \Sigma \nvdash X \to Y \Rightarrow \exists r.(r \models \Sigma \wedge r \not\models X \to Y)
\end{aligned}
$$

The crux is to find an instance r,
with $X^\star = X_1 \ldots X_n$ et $Z_1 \ldots Z_p = R \setminus X^\star$

r	X_1	$\ldots$	X_n	Z_1	$\ldots$	Z_p
s	x_1	$\ldots$	x_n	z_1	$\ldots$	z_p
t	x_1	$\ldots$	x_n	y_1	$\ldots$	y_p

$r \models \Sigma$ but $r \not\models X \to Y$

Functional Dependencies

Inference

Closure algorithm

Normalization

Inference problem for FDs

Armstrong's system leads to a (inefficient) decision procedure for the *inference problem*.

Inference problem for FDs

Let F be a set of FDs and f a single FD, does $F \models f$ hold true?

Lemma: $F \models X \rightarrow Y$ iff $Y \subseteq X^+$

Thus, if we have an (efficient) algorithm to compute X^+, we can (efficiently) solve the inference problem:

1. Given Σ and $X \rightarrow Y$, compute X^+ w.r.t. Σ
2. Return $Y \subseteq X^+$

Closure algorithm: $\mathit{Closure}(\Sigma, X)$

Data: Σ a set of FDs, X a set of d'attributes.
Result: X^+, the closure of X w.r.t. Σ

$\mathit{Cl} := X$
$\mathit{done} := \mathit{false}$
while $(\neg \mathit{done})$ **do**
    $\mathit{done} := \mathit{true}$
    forall $W \to Z \in \Sigma$ **do**
        if $W \subseteq \mathit{Cl} \wedge Z \not\subseteq \mathit{Cl}$ **then**
            $\mathit{Cl} := \mathit{Cl} \cup Z$
            $\mathit{done} := \mathit{false}$
return Cl

Algorithm 1: $\mathit{Closure}(\Sigma, X)$

How many times[3] do we compute $W \subseteq Cl \wedge Z \nsubseteq Cl$ w.r.t. $|\Sigma| = n$?

[3] at worst, using a bad strategy at line 5.

Second algorithm

Data: Σ a set of FDs, X a set of d'attributes.
Result: X^+, the closure of X w.r.t. Σ

for $W \to Z \in F$ **do**
    $count[W \to Z] := |W|$
    for $A \in W$ **do**
        $list[A] := list[A] \cup W \to Z$
$closure := X$, $update := X$
while $update \neq \emptyset$ **do**
    Choose $A \in update$
    $update := update \setminus \{A\}$
    for $W \to Z \in list[A]$ **do**
        $count[W \to Z] := count[W \to Z] - 1$
        if $count[W \to Z] = 0$ **then**
            $update := update \cup (Z \setminus closure)$
            $closure := closure \cup Z$
return $closure$

Algorithm 2: $Closure'(\Sigma, X)$

Example : AE^+

$$\Sigma = \{A \rightarrow I; AB \rightarrow E; BI \rightarrow E; CD \rightarrow I; E \rightarrow C\}$$

Initialization

$$\begin{array}{ll}
List[A] = \{A \rightarrow D; AB \rightarrow E\} & count[A \rightarrow D] = 1 \\
List[B] = \{AB \rightarrow E; BI \rightarrow E\} & count[AB \rightarrow E] = 2 \\
List[C] = \{CD \rightarrow I\} & count[BI \rightarrow E] = 2 \\
List[D] = \{CD \rightarrow I\} & count[CD \rightarrow I] = 2 \\
List[E] = \{E \rightarrow C\} & count[E \rightarrow C] = 1 \\
List[I] = \{BI \rightarrow E\} &
\end{array}$$

Cover

Cover of a set of FDs

With $F^+ = \{f \mid F \models f\}$, let Σ et Γ be two sets of FDs,
Γ is a cover of Σ *iff* $\Gamma^+ = \Sigma^+$

Data: F a set of FDs
Result: G a *minimal* (in cardinality) cover of F

```
G := ∅
for X → Y ∈ F do
    G := G ∪ {X → X⁺};
for X → X⁺ ∈ G do
    if G \ {X → X⁺} ⊢ X → X⁺ then
        G := G \ {X → X⁺};
return G;
```

Algorithm 3: *Minimize*(F)

Normal forms

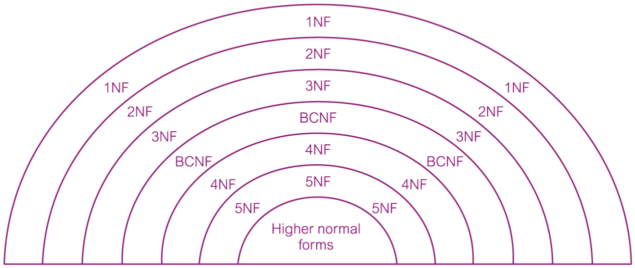

Figure 13.7 Diagrammatic illustration of the relationship between the normal forms.

Application of FD: Normalization

We write $\langle R, \Sigma \rangle$ with R a relation schema and Σ a set of FDs on R.
A set of attribute X is a *minimal key* of $\langle R, \Sigma \rangle$ iff:

- X is a key of R (i.e., $X \to R$ holds)
- X is *minimal* w.r.t. set inclusion: $\forall . X' \subsetneq X \Rightarrow X' \not\to R$

Third Normal Form (3NF)

$\langle R, \Sigma \rangle$ is in 3NF iff, for all *non-trivial* FD $X \to A$ of Σ^+, one of the following conditions holds:

- X is a key of R
- A is a member of *at least* one minimal key of R[4]

Boyce-Codd Normal Form (BCNF)

$\langle R, \Sigma \rangle$ is in BCNF iff, for all *non-trivial* $X \to A$ of Σ^+, X is a key of R.

Informally, $\langle R, \Sigma \rangle$ is good when Σ is nothing but the key!

[4] An attribute that appears in *at least* one minimal key is said to be a *prime attribute*.

Example

3NF captures most of redundancies

- $\langle ABC, \{A \to B, B \to C\}\rangle$ is *not* in 3NF
  A is the unique *minimal* key. Considering $B \to C$, C is *not* prime and B is *not* a key. Clearly, ABC should be divided into AB and BC
- $\langle ABC, \{AB \to C, C \to B\}\rangle$ *is* in 3NF
  There are two *minimal* keys: AB and AC. Every attribute is prime so the 3NF condition holds. Unfortunately, some redundancies still hold but there is no way to decompose ABC into smaller relation without loss of FD!

BCNF captures all redundancies (expressed by FD)

- $\langle ABC, \{AB \to C, C \to B\}\rangle$ is *not* in BCNF
  Considering $C \to B$, C alone is not a key.

Synthesis algorithm

Data: R the set of all attributes
Data: Σ a set of FDs on R
Result: A decomposition $\mathbf{R}$ of R according to Σ

$F := \mathit{Reduce}(\mathit{Minimize}(\Sigma))$
for $X \to Y \in F$ **do**
  $\mathbf{R} := \mathbf{R} \cup \{XY\}$
for $R \in \mathbf{R}$ **do**
  if $\exists R'.R \subsetneq R'$ **then** $\mathbf{R} := \mathbf{R} \setminus \{R\}$;
$\mathit{Keys} := \{X \mid X \to U \wedge \forall Z.Z \subsetneq X \Rightarrow Z \not\to U\}$
if $\forall R \in \mathbf{R}.\ \nexists K \in \mathit{Cle}.K \subseteq R$ **then**
  $\mathit{pick}\ K \in \mathit{Cle}$
  $\mathbf{R} := \mathbf{R} \cup \{K\}$
return $\mathbf{R}$

Algorithm 4: $\mathit{Synthesis}(\Sigma, U)$

End.