Contraintes & poursuite

Cours L3 Bases de Données

Pierre Senellart

27 mars 2019

Plan

Cadre

- Modèle relationnel
- Sémantique ensembliste
- Perspective nommée (pas d'ordre entre attributs)
- Attributs non typés
- Un schéma de relation peut donc être vu comme un ensemble de noms d'attributs
- Tout ceci se généralise facilement à d'autres contextes (sauf sémantique multi-ensembliste, plus complexe)

Dépendance fonctionnelle (FD)

- Relation entre attributs d'une même table R
- Interdit la présence de certains tuples dans la base
- Généralise la notion de clef
- Se note $X \to Y$ où X et Y sont deux ensembles de noms d'attributs d'une même table
- Définition : Une instance r vérifie $X \to Y$, noté $r \models X \to Y$ si $\forall t, t' \in r$, si $t[X] = t'[X]$, alors $t[Y] = t'[Y]$
- Ne peut être spécifié comme contrainte en SQL (c'est normal : si une relation est en BCNF, pas de FD non triviale)

Dépendance multivaluée (MVD)

- Relation entre attributs d'une même table R
- Impose la présence de certains tuples dans la table
- Se note $X \twoheadrightarrow Y$ où X et Y sont deux ensembles de noms d'attributs d'une même table
- Définition : Une instance r vérifie $X \twoheadrightarrow Y$, noté $r \models X \twoheadrightarrow Y$ si $\forall t, t' \in r$, si $t[X] = t'[X]$, et Z est l'ensemble des attributs de R hors $X \cup Y$, alors il existe un tuple t'' tel que $t''[X] = t[X]$, $t''[Y] = t[Y]$ et $t''[Z] = t'[Z]$
- Ne peut être spécifié comme contrainte en SQL (c'est normal : si une relation est en 4NF, pas de MVD non triviale)

À quoi servent les FD et MVD ?

- Comme toutes les contraintes : imposent au niveau du schéma des contraintes logiques sur la base, permettant d'assurer l'intégrité des données
- Permettent de définir une théorie de la normalisation (cf. cours antérieur)
- Cas particuliers importants d'une famille de dépendances (EGD+TGD) apparaissant dans un grand nombre de contextes (voir plus loin)

Plan

La poursuite

Soit R un schéma de relation, et soit Σ un ensemble de dépendances fonctionnelles et multivaluées sur R.

La poursuite (chase) est un algorithme qui décide de si une dépendance fonctionnelle ou multi-valuée est impliquée par Σ dans R :

$$\Sigma \overset{?}{\models} \sigma$$

En d'autres termes :

$$\forall r \text{ instance of } R \quad r \models \Sigma \quad \overset{?}{\Longrightarrow} \quad r \models \sigma$$

Plan

Exemples

On considère le schéma $\{A, B, C, D\}$.

1. $\{\{A\} \twoheadrightarrow \{B, C\}, \{D\} \rightarrow \{C\}\} \stackrel{?}{\models} \{A\} \rightarrow \{C\}$

Exemples

On considère le schéma $\{A, B, C, D\}$.

1. $\{\{A\} \twoheadrightarrow \{B, C\}, \{D\} \rightarrow \{C\}\} \overset{?}{\models} \{A\} \rightarrow \{C\}$
2. $\{\{A\} \twoheadrightarrow \{B\}, \{B\} \twoheadrightarrow \{C\}\} \overset{?}{\models} \{A\} \twoheadrightarrow \{C\}$

Exemples

On considère le schéma $\{A, B, C, D\}$.

1. $\{\{A\} \twoheadrightarrow \{B, C\}, \{D\} \rightarrow \{C\}\} \overset{?}{\models} \{A\} \rightarrow \{C\}$
2. $\{\{A\} \twoheadrightarrow \{B\}, \{B\} \twoheadrightarrow \{C\}\} \overset{?}{\models} \{A\} \twoheadrightarrow \{C\}$
3. $\{\{A\} \twoheadrightarrow \{B, C\}, \{C, D\} \rightarrow \{B\}\} \overset{?}{\models} \{A\} \rightarrow \{B\}$

Exemple 1 (1/6)

$$\{\{A\} \twoheadrightarrow \{B, C\}, \{D\} \rightarrow \{C\}\} \stackrel{?}{\models} \{A\} \rightarrow \{C\}$$

Créer une instance r sur le schéma $\{A, B, C, D\}$ avec deux tuples et des valeurs distinctes pour tous les attributs.

A	B	C	D
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2

Exemple 1 (2/6)

$$\{\{A\} \twoheadrightarrow \{B, C\}, \{D\} \rightarrow \{C\}\} \stackrel{?}{\models} \{A\} \rightarrow \{C\}$$

On veut poursuivre $\{A\} \rightarrow \{C\}$.
Rendre toutes les valeurs de A identiques.

$$a_1 = a_2$$

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2

Exemple 1 (3/6)

$$\{\{A\} \twoheadrightarrow \{B, C\}, \{D\} \rightarrow \{C\}\} \stackrel{?}{\models} \{A\} \rightarrow \{C\}$$

Utiliser $\{A\} \twoheadrightarrow \{B, C\}$. Créer deux tuples en copiant les deux tuples avec la même valeur de A mais en échangeant leurs valeurs pour $\{B, C\}$. La dépendance multivaluée génère des tuples. C'est une dépendance génératrice de tuples (tuple-generating dependency, TGD).

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2
a_1	b_2	c_2	d_1
a_1	b_1	c_1	d_2

Exemple 1 (4/6)

$$\{\{A\} \twoheadrightarrow \{B, C\}, \{D\} \rightarrow \{C\}\} \stackrel{?}{\models} \{A\} \rightarrow \{C\}$$

Utiliser $\{D\} \rightarrow \{C\}$. Pour chaque deux tuples avec la même valeur de D, rendre leur valeur de C identique : $c_1 = c_2$. La dépendance fonctionnelle engendre des égalités. C'est une dépendance génératrice d'égalités (equality-generating dependency, EGD).

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_1	b_2	c_1	d_1
a_1	b_1	c_1	d_2

Exemple 1 (5/6)

$$\{\{A\} \twoheadrightarrow \{B, C\}, \{D\} \rightarrow \{C\}\} \stackrel{?}{\models} \{A\} \rightarrow \{C\}$$

Aucune règle ne s'applique. On observe que r vérifie $\{A\} \rightarrow \{C\}$.

$$r \models \{A\} \rightarrow \{C\}$$

La réponse est donc oui.

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_1	b_2	c_1	d_1
a_1	b_1	c_1	d_2

Exemple 1 (6/6)

r satisfait également $\{D\} \rightarrow \{A\}$ mais c'est une coïncidence.
On ne peut répondre qu'à la question à propos de $\{A\} \rightarrow \{C\}$.
Une autre poursuite serait nécessaire pour $\{D\} \rightarrow \{A\}$.

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_1	b_2	c_1	d_1
a_1	b_1	c_1	d_2

Exemple 2 (1/5)

$$R = \{A, B, C, D\}$$

$$\{\{A\} \twoheadrightarrow \{B\}, \{B\} \twoheadrightarrow \{C\}\} \overset{?}{\models} \{A\} \twoheadrightarrow \{C\}$$

A	B	C	D
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2

Exemple 2 (2/5)

$$R = \{A, B, C, D\}$$

$$\{\{A\} \twoheadrightarrow \{B\}, \{B\} \twoheadrightarrow \{C\}\} \overset{?}{\models} \{A\} \twoheadrightarrow \{C\}$$

On veut poursuivre $\{A\} \twoheadrightarrow \{C\}$.
On rend toutes les valeurs de A identiques.

$$a_1 = a_2$$

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2

Exemple 2 (3/5)

$$R = \{A, B, C, D\}$$

$$\{\{A\} \twoheadrightarrow \{B\}, \{B\} \twoheadrightarrow \{C\}\} \overset{?}{\models} \{A\} \twoheadrightarrow \{C\}$$

Utiliser $\{A\} \twoheadrightarrow \{B\}$.

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2
a_1	b_2	c_1	d_1
a_1	b_1	c_2	d_2

Exemple 2 (4/5)

$$R = \{A, B, C, D\}$$

$$\{\{A\} \twoheadrightarrow \{B\}, \{B\} \twoheadrightarrow \{C\}\} \overset{?}{\models} \{A\} \twoheadrightarrow \{C\}$$

Utiliser $\{B\} \twoheadrightarrow \{C\}$ (deux fois).

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2
a_1	b_2	c_1	d_1
a_1	b_1	c_2	d_2
a_1	b_1	c_2	d_1
a_1	b_1	c_1	d_2
a_1	b_2	c_1	d_2
a_1	b_2	c_2	d_1

Exemple 2 (5/5)

Aucune règle ne s'applique.

$$r \models \{A\} \twoheadrightarrow \{C\}$$

La réponse est donc oui.

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2
a_1	b_2	c_1	d_1
a_1	b_1	c_2	d_2
a_1	b_1	c_2	d_1
a_1	b_1	c_1	d_2
a_1	b_2	c_1	d_2
a_1	b_2	c_2	d_1

Exemple 3 (1/3)

$$\{\{A\} \twoheadrightarrow \{B, C\}, \{C, D\} \rightarrow \{B\}\} \stackrel{?}{\models} \{A\} \rightarrow \{B\}$$

A	B	C	D
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2

Exemple 3 (2/3)

$$\{\{A\} \twoheadrightarrow \{B,C\}, \{C,D\} \rightarrow \{B\}\} \stackrel{?}{\models} \{A\} \rightarrow \{B\}$$

Utiliser $\{A\} \twoheadrightarrow \{B,C\}$.

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2
a_1	b_2	c_2	d_1
a_1	b_1	c_1	d_2

Exemple 3 (3/3)

Aucune règle ne s'applique.

$$r \not\models \{A\} \rightarrow \{B\}$$

La réponse est donc non.

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2
a_1	b_2	c_2	d_1
a_1	b_1	c_1	d_2

Nous avons construit un contre-exemple.

Plan

Le pouvoir de la poursuite

La poursuite est un outil très puissant : il peut être utilisé pour prouver qu'une dépendance fonctionnelle ou multivaluée est vérifiée !

Le pouvoir de la poursuite

La poursuite est un outil très puissant : il peut être utilisé pour prouver qu'une dépendance fonctionnelle ou multivaluée est vérifiée !

Théorème ([Maier et al., 1979])

La poursuite par des FD et MVD termine toujours après un nombre fini d'opérations, et construit un contre-exemple si et seulement s'il en existe un.

Le pouvoir de la poursuite

La poursuite est un outil très puissant : il peut être utilisé pour prouver qu'une dépendance fonctionnelle ou multivaluée est vérifiée !

Théorème ([Maier et al., 1979])

La poursuite par des FD et MVD termine toujours après un nombre fini d'opérations, et construit un contre-exemple si et seulement s'il en existe un.

Pourquoi ? On verra que c'est un cas particulier d'un résultat beaucoup plus général.

Démarrer la poursuite

Soit Σ un ensemble de dépendances fonctionnelles et multivaluées sur un schéma de relation R. Soit σ une dépendance fonctionnelle ou multivaluée.

$$\sigma = X \to Y \text{ or } \sigma = X \twoheadrightarrow Y$$

Démarrer la poursuite

Soit Σ un ensemble de dépendances fonctionnelles et multivaluées sur un schéma de relation R. Soit σ une dépendance fonctionnelle ou multivaluée.

$$\sigma = X \to Y \text{ or } \sigma = X \twoheadrightarrow Y$$

1. Créer une table r de schéma R avec deux tuples avec des valeurs toutes différentes.

Démarrer la poursuite

Soit Σ un ensemble de dépendances fonctionnelles et multivaluées sur un schéma de relation R. Soit σ une dépendance fonctionnelle ou multivaluée.

$$\sigma = X \to Y \text{ or } \sigma = X \twoheadrightarrow Y$$

1. Créer une table r de schéma R avec deux tuples avec des valeurs toutes différentes.
2. Pour chaque $A \in X$, rendre les valeurs de A identiques (en choisissant des valeurs différentes pour chaque A).

Poursuite

Répéter jusqu'à obtenir un point fixe :

1. Pour chaque dépendance fonctionnelle $Z \to V \in \Sigma$:
   - Si deux tuples ont la même valeur de Z, rendre leur valeur de V identique.
2. Pour chaque dépendance multivaluée $Z \twoheadrightarrow V \in \Sigma$:
   - Si deux tuples ont la même valeur de Z, ajouter deux nouveaux tuples avec les mêmes valeurs, sauf que les valeurs de V sont échangées.

Au point fixe (qu'on obtient toujours) :

$$r \models \sigma \quad \text{est équivalent à} \quad \Sigma \models \sigma$$

Il suffit de vérifier si r vérifie σ.

Plan

Tester si une décomposition est sans perte

Si $R = X \cup Y \cup Z$ avec X, Y, Z distincts, une MVD $X \twoheadrightarrow Y$ est vérifiée par r ssi :

$$\pi_{X \cup Y}(r) \bowtie \pi_{X \cup Z}(r) = r$$

On peut donc utiliser les MVD pour tester si un schéma de relation $R = X \cup Y \cup Z$ (satisfaisant des dépendances Σ) peut être décomposé en deux relations $R_1 = X \cup Y$ et $R_2 = X \cup Z$ sans perte :

Tester si une décomposition est sans perte

Si $R = X \cup Y \cup Z$ avec X, Y, Z distincts, une MVD $X \twoheadrightarrow Y$ est vérifiée par r ssi :

$$\pi_{X \cup Y}(r) \bowtie \pi_{X \cup Z}(r) = r$$

On peut donc utiliser les MVD pour tester si un schéma de relation $R = X \cup Y \cup Z$ (satisfaisant des dépendances Σ) peut être décomposé en deux relations $R_1 = X \cup Y$ et $R_2 = X \cup Z$ sans perte :

Utiliser la poursuite pour décider si $\Sigma \overset{?}{\models} X \twoheadrightarrow Y$

(ou, de manière équivalente, $\Sigma \overset{?}{\models} X \twoheadrightarrow Z$)

Plan

Plan

FD comme dépendances logiques

$R = (A_1, \ldots A_n)\,;\, \{A_1\} \rightarrow \{A_2 A_3\}$ peut s'exprimer :

$$\forall a_1 \forall a_2 \forall a_3 \ldots \forall a_n \forall a'_2 \forall a'_3 \ldots \forall a'_n$$

$$R(a_1, a_2, a_3, a_4, \ldots, a_n) \wedge R(a_1, a'_2, a'_3, a'_4, \ldots, a'_n) \Longrightarrow a_2 = a'_2 \wedge a_3 = a'_3$$

FD comme dépendances logiques

$R = (A_1, \ldots A_n)\,; \{A_1\} \rightarrow \{A_2 A_3\}$ peut s'exprimer :

$$\forall a_1 \forall a_2 \forall a_3 \ldots \forall a_n \forall a'_2 \forall a'_3 \ldots \forall a'_n$$
$$R(a_1, a_2, a_3, a_4, \ldots, a_n) \wedge R(a_1, a'_2, a'_3, a'_4, \ldots, a'_n) \Longrightarrow a_2 = a'_2 \wedge a_3 = a'_3$$

Une telle formule est une instance d'une formule plus générale :

$$\forall x_1 \ldots \forall x_m \quad \varphi(x_1 \ldots x_m) \Rightarrow \psi(x_1 \ldots x_m)$$

où $\varphi(x_1 \ldots x_m)$ est une CQ avec variables $x_1 \ldots x_m$ et
$\psi(x_1 \ldots x_m)$ une conjonction d'égalités avec variables $x_1 \ldots x_m$.

FD comme dépendances logiques

$R = (A_1, \ldots A_n)\,; \{A_1\} \rightarrow \{A_2 A_3\}$ peut s'exprimer :

$$\forall a_1 \forall a_2 \forall a_3 \ldots \forall a_n \forall a'_2 \forall a'_3 \ldots \forall a'_n$$
$$R(a_1, a_2, a_3, a_4, \ldots, a_n) \wedge R(a_1, a'_2, a'_3, a'_4, \ldots, a'_n) \Longrightarrow a_2 = a'_2 \wedge a_3 = a'_3$$

Une telle formule est une instance d'une formule plus générale :

$$\forall x_1 \ldots \forall x_m \quad \varphi(x_1 \ldots x_m) \Rightarrow \psi(x_1 \ldots x_m)$$

où $\varphi(x_1 \ldots x_m)$ est une CQ avec variables $x_1 \ldots x_m$ et $\psi(x_1 \ldots x_m)$ une conjonction d'égalités avec variables $x_1 \ldots x_m$.
De telles formules sont appelées des dépendances génératrices d'égalités (EGD).
Les FD ont la particularité d'avoir :

- seulement 2 faits dans la partie gauche
- seulement 1 relation mentionnée

MVDs as logical dependencies

$R = (A_1, \ldots A_n)\,;\, \{A_1\} \twoheadrightarrow \{A_2 A_3\}$ peut s'exprimer :

$$\forall a_1 \forall a_2 \forall a_3 \ldots \forall a_n \forall a'_2 \forall a'_3 \ldots \forall a'_n$$

$$R(a_1, a_2, a_3, \ldots, a_n) \wedge R(a_1, a'_2, a'_3, \ldots, a'_n) \Longrightarrow R(a_1, a'_2, a'_3, a_4, \ldots, a_n)$$

MVDs as logical dependencies

$R = (A_1, \ldots A_n)\,; \{A_1\} \twoheadrightarrow \{A_2 A_3\}$ peut s'exprimer :

$$\forall a_1 \forall a_2 \forall a_3 \ldots \forall a_n \forall a'_2 \forall a'_3 \ldots \forall a'_n$$

$$R(a_1, a_2, a_3, \ldots, a_n) \wedge R(a_1, a'_2, a'_3, \ldots, a'_n) \Longrightarrow R(a_1, a'_2, a'_3, a_4, \ldots, a_n)$$

Une telle formule est une instance d'une formule plus générale :

$$\forall x_1 \ldots \forall x_m \quad \varphi(x_1 \ldots x_m) \Rightarrow \exists y_1 \ldots \exists y_p \; \psi(x_1 \ldots x_m, y_1, \ldots, y_p)$$

où $\varphi(x_1 \ldots x_m)$ est une CQ avec variables $x_1 \ldots x_m$ et $\psi(x_1 \ldots x_m, y_1, \ldots, y_p)$ une CQ avec variables $x_1 \ldots x_m, y_1 \ldots y_p$.

MVDs as logical dependencies

$R = (A_1, \ldots A_n)\,; \{A_1\} \twoheadrightarrow \{A_2 A_3\}$ peut s'exprimer :

$$\forall a_1 \forall a_2 \forall a_3 \ldots \forall a_n \forall a'_2 \forall a'_3 \ldots \forall a'_n$$
$$R(a_1, a_2, a_3, \ldots, a_n) \wedge R(a_1, a'_2, a'_3, \ldots, a'_n) \Longrightarrow R(a_1, a'_2, a'_3, a_4, \ldots, a_n)$$

Une telle formule est une instance d'une formule plus générale :

$$\forall x_1 \ldots \forall x_m \quad \varphi(x_1 \ldots x_m) \Rightarrow \exists y_1 \ldots \exists y_p \; \psi(x_1 \ldots x_m, y_1, \ldots, y_p)$$

où $\varphi(x_1 \ldots x_m)$ est une CQ avec variables $x_1 \ldots x_m$ et $\psi(x_1 \ldots x_m, y_1, \ldots, y_p)$ une CQ avec variables $x_1 \ldots x_m, y_1 \ldots y_p$.
De telles formules sont appelées des dépendances génératrices de tuples (TGD).
Les MVD ont la particularité d'avoir :

- seulement 2 faits dans la partie gauche (plus de 2 : JD)
- seulement 1 fait dans la partie droite, pas de $\exists$
- seulement une relation mentionnée

Plan

Dépendances d'inclusion

- Les FD et MVD ne sont capables d'exprimer des contraintes (ou dépendances) que locales à une relation

Dépendances d'inclusion

- Les FD et MVD ne sont capables d'exprimer des contraintes (ou dépendances) que locales à une relation
- En pratique, on veut exprimer des dépendances entre relations, en particulier des clefs étrangères

Dépendances d'inclusion

- Les FD et MVD ne sont capables d'exprimer des contraintes (ou dépendances) que locales à une relation
- En pratique, on veut exprimer des dépendances entre relations, en particulier des clefs étrangères
- Les tuples des valeurs de certains attributs sont inclus dans les valeurs d'autres attributs (par exemple d'une autre table)

Dépendances d'inclusion

- Les FD et MVD ne sont capables d'exprimer des contraintes (ou dépendances) que locales à une relation
- En pratique, on veut exprimer des dépendances entre relations, en particulier des clefs étrangères
- Les tuples des valeurs de certains attributs sont inclus dans les valeurs d'autres attributs (par exemple d'une autre table)
- En SQL :

```
CREATE TABLE R1(
  A INT, B VARCHAR(42), ...,
  FOREIGN KEY (A,B) REFERENCES R2(X,Y) )
```

Dépendances d'inclusion

- Les FD et MVD ne sont capables d'exprimer des contraintes (ou dépendances) que locales à une relation
- En pratique, on veut exprimer des dépendances entre relations, en particulier des clefs étrangères
- Les tuples des valeurs de certains attributs sont inclus dans les valeurs d'autres attributs (par exemple d'une autre table)
- En SQL :

```
CREATE TABLE R1(
  A INT, B VARCHAR(42), ...,
  FOREIGN KEY (A,B) REFERENCES R2(X,Y) )
```

- Notation : $R_1[A, B] \subseteq R_2[C, D]$

Les ID comme dépendances logiques

$R = (A_1, \ldots A_n)$, $S = (B_1, \ldots B_m)$ $R[A_1] \subseteq S[B_2]$ peut s'exprimer :

$$\forall a_1 \forall a_2 \forall a_3 \ldots \forall a_n$$

$$R(a_1, a_2, a_3, \ldots, a_n) \Longrightarrow \exists b_1 \exists b_3 \ldots \exists b_m \; S(b_1, a_1, b_3, \ldots, b_m)$$

Les ID comme dépendances logiques

$R = (A_1, \ldots A_n)$, $S = (B_1, \ldots B_m)$ $R[A_1] \subseteq S[B_2]$ peut s'exprimer :

$\forall a_1 \forall a_2 \forall a_3 \ldots \forall a_n$

$$R(a_1, a_2, a_3, \ldots, a_n) \Longrightarrow \exists b_1 \exists b_3 \ldots \exists b_m \; S(b_1, a_1, b_3, \ldots, b_m)$$

Une telle formule est une instance de TGD :

$$\forall x_1 \ldots \forall x_m \quad \varphi(x_1 \ldots x_m) \Rightarrow \exists y_1 \ldots \exists y_p \; \psi(x_1 \ldots x_m, y_1, \ldots, y_p)$$

Les ID ont la particularité d'avoir :

- seulement 1 fait du côté gauche
- seulement 1 fait du côté droit

Plan

Poursuite générale

- La poursuite ne se limite pas qu'aux FD et MVD
- En fait, peut être définie pour un ensemble arbitraire de TGD et EGD, qui va au-delà des FD, MVD, JD, et même des ID
- Les TGD (et les EGD) sont un objet fondamental de recherche, au-delà des contraintes sur les bases de données. Connu dans d'autres communautés sous le nom de *règles existentielles* [Baget et al., 2011], *Datalog*$^{\pm}$ [Calì et al., 2010]
- La poursuite est un outil majeur, en pratique et en théorie, pour travailler avec ces objets

Démarrer la poursuite

Soit Σ un ensemble de TGD et EGD sur un schéma relationnel $\mathcal{S}$. Soit σ une TGD ou EGD.

Démarrer la poursuite

Soit Σ un ensemble de TGD et EGD sur un schéma relationnel $\mathcal{S}$. Soit σ une TGD ou EGD.

1. Créer une base de données d de schéma $\mathcal{S}$ avec comme tuples les atomes apparaissant dans la partie gauche de σ.

Démarrer la poursuite

Soit Σ un ensemble de TGD et EGD sur un schéma relationnel $\mathcal{S}$. Soit σ une TGD ou EGD.

1. Créer une base de données d de schéma $\mathcal{S}$ avec comme tuples les atomes apparaissant dans la partie gauche de σ.
2. S'assurer que les variables partagées entre ces atomes de σ ont la même valeur dans la bases de données initiale (et les variables non partagées des valeurs distinctes).

Poursuite

Répéter jusqu'à obtenir un point fixe :

1. Pour chaque EGD :
   - Si la CQ de la partie gauche est satisfaite dans la base, imposer les égalités de la partie droite sur les valeurs de la base.
2. Pour chaque TGD :
   - Si la CQ de la partie gauche est satisfaite dans la base, mais pas celle de la partie droite, ajouter à la base les tuples indiquées par la partie droite. Pour les variables quantifiées existentiellement, introduire de nouvelles valeurs (toujours fraîches, jamais utilisées).

Au point fixe (qu'on peut ne pas atteindre) :

$$r \models \sigma \text{ est équivalent à } \Sigma \models \sigma$$

Quand a-t-on terminaison ?

Théorème

S'il n'y a pas de variable existentiellement quantifiée dans les dépendances, la poursuite termine.

Quand a-t-on terminaison ?

Théorème

S'il n'y a pas de variable existentiellement quantifiée dans les dépendances, la poursuite termine.

Démonstration.

Comme on ne peut introduire de nouvelles valeurs, la poursuite ne peut produire qu'un nombre borné de tuples. □

Quand a-t-on terminaison ?

Théorème

S'il n'y a pas de variable existentiellement quantifiée dans les dépendances, la poursuite termine.

Démonstration.

Comme on ne peut introduire de nouvelles valeurs, la poursuite ne peut produire qu'un nombre borné de tuples. □

Peut être généralisé à d'autres cadres : quand les TGD n'introduisent pas de cycles, ou qu'ils n'introduisent que des cycle « gentils » (p. ex., acyclicité faible [Fagin et al., 2005]). Mais échoue si on a des FD et ID arbitraires.

Quand a-t-on terminaison ?

Théorème

S'il n'y a pas de variable existentiellement quantifiée dans les dépendances, la poursuite termine.

Démonstration.

Comme on ne peut introduire de nouvelles valeurs, la poursuite ne peut produire qu'un nombre borné de tuples. □

Peut être généralisé à d'autres cadres : quand les TGD n'introduisent pas de cycles, ou qu'ils n'introduisent que des cycle « gentils » (p. ex., acyclicité faible [Fagin et al., 2005]). Mais échoue si on a des FD et ID arbitraires. En fait, l'implication d'une dépendance par une collection de FD et ID est indécidable. [Chandra et al., 1981]

Pourquoi est-ce que la poursuite est importante ?

Théorème ([Fagin et al., 2005])

Le résultat de la poursuite est un modèle universel : toutes les bases de données satisfaisant les dépendances ont une sous-instance qui est homomorphe au résultat de la poursuite. Autrement dit, une CQ est vraie sur le résultat de la poursuite si et seulement si elle est vraie sur toutes les bases de données satisfaisant les dépendances.

Pourquoi est-ce que la poursuite est importante ?

Théorème ([Fagin et al., 2005])

Le résultat de la poursuite est un modèle universel : toutes les bases de données satisfaisant les dépendances ont une sous-instance qui est homomorphe au résultat de la poursuite. Autrement dit, une CQ est vraie sur le résultat de la poursuite si et seulement si elle est vraie sur toutes les bases de données satisfaisant les dépendances.

- Noter que le résultat de la poursuite n'est pas unique (dépend de l'ordre d'application), mais fonctionne pour n'importe quel résultat
- Quand la poursuite ne termine pas, peut tout de même être utile : p. ex., on peut parfois se limiter à une portion bornée de la poursuite

Autres applications

- Répondre aux requêtes sur toutes les complétions possibles d'une bases de données contraintes par des TGD et EGD [Calì et al., 2003, 2012] ; beaucoup utilisé dans les logiques du Web sémantique.
- Intégration de données [Lenzerini, 2002] : les TGD et EGD expriment les relations entre deux schémas, et on effectue des requêtes sur un jeu de données basé sur les dépendances et un jeu de données de l'autre schéma.
- Échange de données [Fagin et al., 2005, Onet, 2013] : les TGD et les EGD expriment les relations entre deux schémas, et on utilise la poursuite pour générer la traduction d'un jeu de données dans l'autre schéma.
- Programme Datalog : ensemble de TGD. On peut raisonner sur Datalog + contraintes en utilisant la poursuite.

Plan

Références

- L'article historique sur la poursuite [Maier et al., 1979]
- Chapitres 8 à 10 de [Abiteboul et al., 1995]
- Une vision moderne de la poursuite générale : [Fagin et al., 2005]

Bibliographie I

Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. ISBN 0-201-53771-0. URL `http://www-cse.ucsd.edu/users/vianu/book.html`.

Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. On rules with existential variables : Walking the decidability line. *Artif. Intell.*, 175(9-10) :1620–1654, 2011. doi : 10.1016/j.artint.2011.03.002. URL `http://dx.doi.org/10.1016/j.artint.2011.03.002`.

Andrea Calì, Domenico Lembo, and Riccardo Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In Frank Neven, Catriel Beeri, and Tova Milo, editors, *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, pages 260–271. ACM,

Bibliographie II

2003. ISBN 1-58113-670-6. doi : 10.1145/773153.773179. URL http://doi.acm.org/10.1145/773153.773179.

Andrea Calì, Georg Gottlob, Thomas Lukasiewicz, and Andreas Pieris. Datalog+/- : A family of languages for ontology querying. In Oege de Moor, Georg Gottlob, Tim Furche, and Andrew Jon Sellers, editors, *Datalog Reloaded - First International Workshop, Datalog 2010, Oxford, UK, March 16-19, 2010. Revised Selected Papers*, volume 6702 of *Lecture Notes in Computer Science*, pages 351–368. Springer, 2010. ISBN 978-3-642-24205-2. doi : 10.1007/978-3-642-24206-9_20. URL http://dx.doi.org/10.1007/978-3-642-24206-9_20.

Bibliographie III

Andrea Calì, Georg Gottlob, and Andreas Pieris. Towards more expressive ontology languages : The query answering problem. *Artif. Intell.*, 193 :87–128, 2012. doi : 10.1016/j.artint.2012.08.002. URL `http://dx.doi.org/10.1016/j.artint.2012.08.002`.

Ashok K. Chandra, Harry R. Lewis, and Johann A. Makowsky. Embedded implicational dependencies and their inference problem. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing, May 11-13, 1981, Milwaukee, Wisconsin, USA*, pages 342–354. ACM, 1981. doi : 10.1145/800076.802488. URL `http://doi.acm.org/10.1145/800076.802488`.

Bibliographie IV

Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange : semantics and query answering. *Theor. Comput. Sci.*, 336(1) :89–124, 2005. doi : 10.1016/j.tcs.2004.10.033. URL `http://dx.doi.org/10.1016/j.tcs.2004.10.033`.

Maurizio Lenzerini. Data integration : A theoretical perspective. In Lucian Popa, Serge Abiteboul, and Phokion G. Kolaitis, editors, *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, pages 233–246. ACM, 2002. ISBN 1-58113-507-6. doi : 10.1145/543613.543644. URL `http://doi.acm.org/10.1145/543613.543644`.

Bibliographie V

David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 4(4) :455–469, 1979. doi : 10.1145/320107.320115. URL `http://doi.acm.org/10.1145/320107.320115`.

Adrian Onet. The chase procedure and its applications in data exchange. In Phokion G. Kolaitis, Maurizio Lenzerini, and Nicole Schweikardt, editors, *Data Exchange, Integration, and Streams*, volume 5 of *Dagstuhl Follow-Ups*, pages 1–37. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013. ISBN 978-3-939897-61-3. doi : 10.4230/DFU.Vol5.10452.1. URL `http://dx.doi.org/10.4230/DFU.Vol5.10452.1`.