Systèmes d'information décisionnels (Data Warehouse)

Dr Dendani-Hadiby Nadjette
Université Badji Mokhtar Annaba
Département d'Informatique
n_dendani@yahoo.fr

2021/2022

Base de donnée déductive

Concept et Motivation

L'idée est de coupler une base de donnée à un ensemble de règles logiques qui permettent d'en déduire de l'information.

Cela pourrait se faire dans le contexte d'un langage de programmation, mais on souhaite un couplage plus direct qui permet de manipuler les données au niveau de la base de données.

Les questions qui se posent sont les suivantes.

- Quelles est la forme des règles que l'on souhaite utiliser ? Comment les interpréter ?
- Les requêtes expressibles en SQL peuvent-elles être exprimées par des règles ?
- Les règles peuvent-elles exprimer plus ?

Définitions préliminaires

Une *base de données déductive (BDD)* est constituée

- d'un ensemble de prédicats (relations), dits *de base* ou *extensionnels*, dont l'extension est conservée explicitement dans la base de données : la *base de données extensionnelle*
- d'un ensemble de prédicats (relations), dits *dérivés* ou *intentionnels*, dont l'extension est définie par des règles déductives : la *base de données intentionnelle*

Exemple :

Base de données extensionnelle : Prédicat *parent* à 2 arguments (ou relation *parent* à 2 attributs)

parent	
anna	*bob*
bob	*chris*
bob	*dan*
dan	*eric*

Base de données intentionnelle : Prédicat *grandparent* à 2 arguments (ou relation *grandparent* à 2 attributs)

grandparent(X,Y) ← *parent(X,Z)* `AND` *parent (Z,Y)*

Le langage des règles déductives : Datalog

Syntaxe

- Un *terme* est soit une variable, soit une constante.

  Exemples :
  - X, Y, Z, ...
  - *anna*, *bob*, *chris*, *dan*, *eric*, ...

- Une *formule atomique* ou un *atome* est un *symbole prédicatif* appliqué à une liste d'arguments qui sont soit des variables, soit des constantes, i.e., des termes.

  Un *symbole prédicatif* peut être
  - Un nom de prédicat de base (extensionnel),
  - Un prédicat dérivé (intentionnel),
  - Une relation arithmétique ($=$, $\leq$, ...).

Exemples :

- Symboles prédicatifs : *parent* (extensionnel), *grandparent* (intentionnel), ...
- Atomes : *parent(bob,chris)*, *parent(X,Z)*, *grandparent(X,Y)*, $X \leq Y$, ...

– Une *règle (déductive)* est une formule de la forme

$$q(\overline{t}) \leftarrow [\texttt{NOT}]\ p_1(\overline{t_1})\ \texttt{AND}\ \ldots\ \texttt{AND}\ [\texttt{NOT}]\ p_n(\overline{t_n})$$

où $q(\overline{t})$ et $p_1(\overline{t_1}), \ldots, p_n(\overline{t_n})$ sont des formules atomiques.

- $q(\overline{t})$ est la *tête* de la règle (ce qui est défini)
- $[\texttt{NOT}]\ p_1(\overline{t_1})\ \texttt{AND}\ \ldots\ \texttt{AND}\ [\texttt{NOT}]\ p_n(\overline{t_n})$ est le *corps* de la règle (la condition de définition)

Exemple :

grandparent(X,Y) ← *parent(X,Z)* AND *parent(Z,Y)*

Restrictions sur les règles déductives

$$q(\overline{t}) \leftarrow [\texttt{NOT}]\ p_1(\overline{t_1})\ \texttt{AND}\ \ldots\ \texttt{AND}\ [\texttt{NOT}]\ p_n(\overline{t_n})$$

1. Les prédicats apparaissant dans la tête (membre de gauche) de règles sont uniquement des prédicats *dérivés* (intentionnels).

   Donc, les règles ne servent pas à définir des prédicats extensionnels, mais uniquement des prédicats intentionnels.
2. *Conditions de sûreté* : toute variable utilisée dans une règle doit apparaître dans au moins un atome dont le prédicat représente une relation et qui n'est pas précédé d'une négation.

   La sûreté a pour but de garantir que l'évaluation des règles est possible.

Exemples :

- Règle non sûre : $q(X,Y) \leftarrow p(X,Z)\ \texttt{AND NOT}\ r(X,Y,Z)\ \texttt{AND}\ X \geq Y$
- Règle sûre : $q(X) \leftarrow p(X)\ \texttt{AND}\ X \geq 0$

L'interprétation des règles datalog

La question est de savoir comment calculer l'extension des prédicats intentionnels définis par des règles datalog. On procède comme suit.

1. Si un prédicat intentionnel est la tête de plusieurs règles, son extension est l'union des extensions données par les différentes règles.
2. Pour chaque règle, on considère tous les tuples des relations extensionnelles non précédées d'une négation figurant dans le corps de la règle. Pour chaque combinaison de tuples, si
   - les tuples sont compatibles avec les occurrences des prédicats extensionnels (attributs pour lesquels une valeur constante est donnée ayant cette valeur),
   - les prédicats relationnels niés et les prédicats arithmétiques sont safistait,

   on ajoute le tuple défini par la règle.

La sûreté garantit que l'on a ainsi pris en compte toutes les valeurs possibles.

L'interprétation des règles datalog : exemple

Supposons que la relation *parent* ait l'extension suivante :

parent	
anna	*bob*
bob	*chris*
bob	*dan*
dan	*eric*

L'évaluation de la règle

grandparent(X,Y) ← *parent(X,Z)* `AND` *parent(Z,Y)*

donne la relation

grandparent	
anna	*chris*
anna	*dan*
bob	*eric*

De l'algègre relationnelle à datalog

Les opérations de l'algèbre relationnelle peuvent aisément se coder en datalog.

$r \cup s$: $rus(\overline{t_1}) \leftarrow r(\overline{t_1})$
$rus(\overline{t_1}) \leftarrow s(\overline{t_1})$

$r - s$: $rms(\overline{t_1}) \leftarrow r(\overline{t_1})$ AND NOT $s(\overline{t_1})$

$r \times s$: $rts(\overline{t_1}, \overline{t_2}) \leftarrow r(\overline{t_1})$ AND $s(\overline{t_2})$

$r \bowtie s$: $rjs(\overline{t_1} \cup \overline{t_2}) \leftarrow r(\overline{t_1})$ AND $s(\overline{t_2})$

$\pi_X r$: $pr(\overline{t_1} \cap X) \leftarrow r(\overline{t_1})$

$\sigma_C r$: $sr(\overline{t_1}) \leftarrow r(\overline{t_1})$ AND $C(\overline{t_1})$

Pour traduire une expression complexe de l'algèbre relationnelle, on définit un prédicat intensionnel par sous-formule de l'expression.

Structure des règles et récursivité

Pour un ensemble de règles donné, le *graphe des règles* est défini comme suit.

- Il y a un nœud dans le graphe pour chaque symbole prédicatif.
- Il existe un arc d'un nœud p à un nœud q ssi il existe une règle de la forme $q(\cdots) \leftarrow \ldots p(\cdots) \ldots$

Un ensemble de règles est *récursif* si son graphe comporte un cycle.

Exemple :

```
ancetre(X,Y)  ←  parent(X,Y)
ancetre(X,Y)  ←  parent(X,Z) AND ancetre(Z,Y)
```

Si un ensemble de règles est récursif, on est amené à utiliser un prédicat intentionnel dans une règle qui le définit. Comment faire pour évaluer ce prédicat ?

L'évaluation des règles récursives

- Le principe utilisé pour *évaluer* les règles récursives est de ne mettre dans l'extension des prédicats intentionels récursifs que les tuples qui doivent y figurer.
- On commence donc avec une extension vide pour ces prédicats, et l'on applique les règles jusqu'à ce que l'on obtienne une extension stable pour les règles. C'est ce que l'on appelle un *point fixe* (*fixpoint*).
- Cette approche est possible, pour autant que les prédicats définis récursivement ne soient pas dans la portée d'un opérateur `NOT`.

L'évaluation des règles récursives : exemple

Considérons le prédicat extensionnel *parent* dont l'extension est

parent	
anna	*bob*
bob	*chris*
bob	*dan*
dan	*eric*

et le prédicate intentionnel *ancêtre* défini par

ancetre(X,Y) ← parent(X,Y)
ancetre(X,Y) ← parent(X,Z) **AND** ancetre(Z,Y)

L'extension calculée pour ancêtre sera successivement

ancêtre	
anna	*bob*
bob	*chris*
bob	*dan*
dan	*eric*

ancêtre	
anna	*bob*
bob	*chris*
bob	*dan*
dan	*eric*
anna	*chris*
anna	*dan*
bob	*eric*

ancêtre	
anna	*bob*
bob	*chris*
bob	*dan*
dan	*eric*
anna	*chris*
anna	*dan*
bob	*eric*
anna	*eric*

Négation et récursion

La négation et la récursion peuvent être utilisées simultanément, pour autant qu'elles n'interagissent pas.

- On impose une restriction : les règles sont *stratifiées*. Si une règle comporte une négation

  $$q(\cdots) \leftarrow \ldots \texttt{NOT}\, p(\cdots) \ldots$$

  il n'y a pas de chemin de q à p dans le graphe des règles.
- Le graphe des règles peut alors être stratifié (divisé en couches) : il n'y a pas de récursion entre couches et dans une règle définissant un prédicat d'une couche, la négation n'est appliquée qu'à des prédicats des couches inférieures.
- L'évaluation de l'extension des prédicats dérivés peut alors se faire couche par couche.

Expressivité

- Les prédicats définis par des règles non récursives peuvent être exprimés en algèbre relationnelle.

- Les prédicats définis par des règles récursives ne peuvent pas toujours être exprimés en algèbre relationnelle.

  Exemple : La fermeture transitive d'une relation (telle que la relation *ancetre*, par exemple). En effet, la calculer nécessite l'application d'un nombre d'opérations qui dépend du contenu de la base de données.

EXEMPLE

- La base de données déductive d'un graphe dirigé contient le prédicat intentionnel **ARC(X,Y)** précisant qu'il existe un arc reliant le sommet **X** au sommet **Y**.

1. Définir le prédicat intentionnel **BOUCLE(X)** qui exprime le fait qu'il existe une suite d'arcs du sommet X vers lui même. Pour cela, il faut définir un prédicat intermédiaire **CHEMIN(X,Y)** qui exprime le fait qu'il existe un chemin entre **X** et **Y**.(arc ou suite d'arcs)
2. Définir l'extension du prédicat **BOUCLE(X)** si l'extension du prédicat **ARC(X,Y)** est celle donnée ci-dessous :

ARC

X1	X2
X2	X3
X2	X4
X3	X2
X2	X1

SOLUTION

1- **CHEMIN(X,Y) <- ARC(X,Y)**
CHEMIN(X,Y) <- ARC(X,Z) AND CHEMIN(Z,Y)
BOUCLE(X) <- CHEMIN(X,X)

2-

ARC

X1	X2
X2	X3
X2	X4
X3	X2
X2	X1

CHEMIN

X1	X2
X2	X3
X2	X4
X3	X2
X2	X1
X1	X3
X1	X4
X3	X4
X2	X2
X3	X3
X1	X1

BOUCLE

X2	X2
X3	X3
X1	X1

FIN