Théorie des bases de données

Pierre Senellart

Colloque *Algorithmique et Programmation*,
Luminy, 11 mai 2018

Théorie des bases de données

- Domaine de recherche bien établi, s'intéressant aux aspects fondamentaux et théoriques de la gestion de données
- Motivations :
  - fournir via des outils théoriques des avancées dans la pratique des gestion de données
  - fournir via un angle données des avancées dans la recherche informatique
- Le succès des SGBD actuels repose en particulier sur un résultat théorique : le théorème de Codd
- Panorama de certains jolis résultats théoriques en BD

Plan

Schéma relationnel

On fixe des ensembles (généralement infinis dénombrables) :

- $\mathcal{L}$ d'étiquettes
- $\mathcal{V}$ de valeurs
- $\mathcal{T}$ de types t.q., $\forall \tau \in \mathcal{T}, \tau \subseteq \mathcal{V}$

Définition

Un schéma de relation (d'arité n) est un n-uplet $(A_1, \ldots, A_n)$ où chaque A_i (appelé attribut) est un couple (L_i, τ_i) avec $L_i \in \mathcal{L}$, $\tau_i \in \mathcal{T}$ et tels que les A_i sont 2 à 2 distincts.

Définition

Un schéma relationnel est défini par un ensemble fini d'étiquettes $L \subseteq \mathcal{L}$ (les noms de relation) chaque étiquette de L étant associée à un schéma de relation.

Exemple de schéma relationnel

- Univers :
  - $\mathcal{L}$ est l'ensemble des chaînes de caractères alphanumériques commençant par une lettre
  - $\mathcal{V}$ est l'ensemble des suites finies de bits
  - $\mathcal{T}$ est formé de types tels que `INTEGER` (représentations sous formes de bits de nombres entiers entre -2^{31} et $2^{31}-1$), `REAL` (représentations de nombres en virgule flottante simple précision IEEE 754), `TEXT` (représentation en UTF-8 de chaînes de caractères), `DATE` (représentation d'une date au format ISO8601), etc.
- Schéma relationnel formé de 2 noms de relations, `Client` et `Reservation`
- `Client` : ((`id`, `INTEGER`), (`nom`, `TEXT`), (`email`, `TEXT`))
- `Reservation` :
  ((`id`, `INTEGER`), (`client`, `INTEGER`), (`chambre`, `INTEGER`), (`arrivee`, `DATE`), (`nuits`, `INTEGER`))

Base de données

Définition

Une instance d'un schéma de relation $((L_1, \tau_1), \ldots, (L_n, \tau_n))$ (on parle aussi d'une relation sur ce schéma) est un ensemble $\{t_1, \ldots, t_k\}$ de n-uplets de la forme $t_j = (v_{j1}, \ldots, v_{jn})$ avec $\forall j \forall i \; v_{ji} \in \tau_i$.

Définition

Une instance d'un schéma relationnel (ou, plus simplement, une base de données sur ce schéma) est une fonction qui à chaque nom de relation associe une instance du schéma de relation correspondant.

Note : On emploie relation de manière ambiguë soit pour un schéma de relation, soit pour une instance d'un schéma de relation.

Exemple

Client

id	nom	email
1	Jean Dupont	jean.dupont@gmail.com
2	Alice Dupuis	alice@dupuis.name
3	Jean Dupont	jean.dupont@ens.fr

Reservation

id	client	chambre	arrivee	nuits
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Quelques notations

- Si $A = (L, \tau)$ est le i-ème attribut d'une relation R, et t un n-uplet d'une instance de R, on note $t[A]$ (ou $t[L]$) la valeur du i-ème composant de t.
- De même, si $\mathcal{A}$ est un k-uplet d'attributs apparaissant parmi les n attributs de R, $t[\mathcal{A}]$ est le k-uplet formé à partir de t en concaténant les $t[A]$ pour A dans $\mathcal{A}$.
- Un tuple est un n-uplet pour un certain n.

Variantes : perspectives nommées et non-nommées

La version présentée considère que les attributs d'une relation sont ordonnés, et ont un nom. C'est ce qui correspond le mieux à ce que font les SGBDR, mais ce n'est pas forcément le plus agréable pour raisonner sur le modèle relationnel.

Perspective nommée. On oublie la position des attributs, on considère qu'ils sont identifiés uniquement par leur nom.

Perspective non-nommée. On oublie le nom des attributs, on considère qu'ils sont identifiés uniquement par leur position. On utilise des notations telles que $t(2)$ pour accéder à la valeur du deuxième attribut d'un tuple.

Pas d'impact majeur, on utilisera l'une ou l'autre perspective suivant ce qui est le plus agréable.

Variante : sémantique multi-ensembliste

- On a défini une instance de relation comme un ensemble de tuples. On peut aussi considérer une sémantique multi-ensembliste du modèle relationnel, où une instance de relation est un multi-ensemble de tuples.
- C'est ce qui correspond le mieux à ce que font les SGBDR...
- ... mais la plupart de la théorie des bases de données relationnelles est fait pour la sémantique ensembliste, plus agréable à manipuler
- On utilisera principalement la sémantique ensembliste dans les résultats théoriques
- Les SGBDR implémentent une sémantique multi-ensembliste, mais on peut forcer une sémantique ensembliste avec le mot-clef **DISTINCT**

Variante : version non typée

- Dans les implémentations, les attributs sont toujours typés
- Dans les modèles et résultats théoriques, on fait souvent abstraction du type des attributs et on considère que chaque attribut a un type universel $\mathcal{V}$
- On omettra donc parfois le type des attributs

Plan

Algèbre relationnelle

- Langage algébrique permettant d'exprimer des requêtes
- Une expression de l'algèbre relationnelle produit une nouvelle relation à partir des relations de la base de données
- Chaque opérateur prend 0, 1, ou 2 sous-expressions
- Principaux opérateurs :

Op.	Arité	Description	Condition
R	0	Nom de relation	$R \in \mathcal{L}$
$\rho_{A \to B}$	1	Renommage	$A, B \in \mathcal{L}$
$\Pi_{A_1 \ldots A_n}$	1	Projection	$A_1 \ldots A_n \in \mathcal{L}$
σ_ϕ	1	Sélection	ϕ formule
$\times$	2	Produit cartésien	
$\cup$	2	Union	
$\setminus$	2	Différence	
$\bowtie_\phi$	2	Jointure	ϕ formule

Nom de relation

Client

id	nom	email
1	Jean Dupont	jean.dupont@gmail.com
2	Alice Dupuis	alice@dupuis.name
3	Jean Dupont	jean.dupont@ens.fr

Reservation

id	client	chambre	arrivee	nuits
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression : `Client`

Résultat :

id	nom	email
1	Jean Dupont	jean.dupont@gmail.com
2	Alice Dupuis	alice@dupuis.name
3	Jean Dupont	jean.dupont@ens.fr

Renommage

Client

id	nom	email
1	Jean Dupont	jean.dupont@gmail.com
2	Alice Dupuis	alice@dupuis.name
3	Jean Dupont	jean.dupont@ens.fr

Reservation

id	client	chambre	arrivee	nuits
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression : $\rho_{\texttt{id}\to\texttt{client}}(\texttt{Client})$

Résultat :

client	nom	email
1	Jean Dupont	jean.dupont@gmail.com
2	Alice Dupuis	alice@dupuis.name
3	Jean Dupont	jean.dupont@ens.fr

Projection

Client

id	nom	email
1	Jean Dupont	jean.dupont@gmail.com
2	Alice Dupuis	alice@dupuis.name
3	Jean Dupont	jean.dupont@ens.fr

Reservation

id	client	chambre	arrivee	nuits
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression : $\Pi_{\texttt{email},\texttt{id}}(\texttt{Client})$

Résultat :

email	id
jean.dupont@gmail.com	1
alice@dupuis.name	2
jean.dupont@ens.fr	3

Sélection

Client

id	nom	email
1	Jean Dupont	jean.dupont@gmail.com
2	Alice Dupuis	alice@dupuis.name
3	Jean Dupont	jean.dupont@ens.fr

Reservation

id	client	chambre	arrivee	nuits
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression : $\sigma_{\texttt{arrivee}>\text{2017-01-12}\wedge\texttt{client}=2}(\texttt{Reservation})$

Résultat :

id	client	chambre	arrivee	nuits
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

La formule utilisée dans la sélection peut être n'importe quelle combinaison booléenne de comparaisons des valeurs d'attributs entre elles et avec des constantes.

Produit cartésien

Client

id	nom	email
1	Jean Dupont	jean.dupont@gmail.com
2	Alice Dupuis	alice@dupuis.name
3	Jean Dupont	jean.dupont@ens.fr

Reservation

id	client	chambre	arrivee	nuits
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression : $\Pi_{\texttt{id}}(\texttt{Client}) \times \Pi_{\texttt{nom}}(\texttt{Client})$

Résultat :

id	nom
1	Alice Dupuis
2	Alice Dupuis
3	Alice Dupuis
1	Jean Dupont
2	Jean Dupont
3	Jean Dupont

Union

Client

id	nom	email
1	Jean Dupont	jean.dupont@gmail.com
2	Alice Dupuis	alice@dupuis.name
3	Jean Dupont	jean.dupont@ens.fr

Reservation

id	client	chambre	arrivee	nuits
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression : $\Pi_{\texttt{chambre}}(\sigma_{\texttt{client}=2}(\texttt{Reservation})) \cup \Pi_{\texttt{chambre}}(\sigma_{\texttt{arrivee}=\text{2017-01-15}}(\texttt{Reservation}))$

Résultat :

chambre
107
302
504

Union

Client

id	nom	email
1	Jean Dupont	jean.dupont@gmail.com
2	Alice Dupuis	alice@dupuis.name
3	Jean Dupont	jean.dupont@ens.fr

Reservation

id	client	chambre	arrivee	nuits
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression : $\Pi_{\texttt{chambre}}(\sigma_{\texttt{client}=2}(\texttt{Reservation})) \cup \Pi_{\texttt{chambre}}(\sigma_{\texttt{arrivee}=\texttt{2017-01-15}}(\texttt{Reservation}))$

Résultat :

chambre
107
302
504

Cette simple union aurait pu être écrite $\Pi_{\texttt{chambre}}(\sigma_{\texttt{client}=2 \vee \texttt{arrivee}=\texttt{2017-01-15}}(\texttt{Reservation}))$.
Pas toujours possible.

Différence

Client

id	nom	email
1	Jean Dupont	jean.dupont@gmail.com
2	Alice Dupuis	alice@dupuis.name
3	Jean Dupont	jean.dupont@ens.fr

Reservation

id	client	chambre	arrivee	nuits
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression : $\Pi_{\texttt{chambre}}(\sigma_{\texttt{client}=2}(\texttt{Reservation})) \setminus \Pi_{\texttt{chambre}}(\sigma_{\texttt{arrivee}=\text{2017-01-15}}(\texttt{Reservation}))$

Résultat :

chambre
107

Différence

Client

id	nom	email
1	Jean Dupont	jean.dupont@gmail.com
2	Alice Dupuis	alice@dupuis.name
3	Jean Dupont	jean.dupont@ens.fr

Reservation

id	client	chambre	arrivee	nuits
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression : $\Pi_{\texttt{chambre}}(\sigma_{\texttt{client}=2}(\texttt{Reservation})) \setminus \Pi_{\texttt{chambre}}(\sigma_{\texttt{arrivee}=\text{2017-01-15}}(\texttt{Reservation}))$

Résultat :

chambre
107

Cette simple différence aurait pu être écrite $\Pi_{\texttt{chambre}}(\sigma_{\texttt{client}=2 \wedge \texttt{arrivee} \neq \text{2017-01-15}}(\texttt{Reservation}))$.

Pas toujours possible.

Jointure

Client

id	nom	email
1	Jean Dupont	jean.dupont@gmail.com
2	Alice Dupuis	alice@dupuis.name
3	Jean Dupont	jean.dupont@ens.fr

Reservation

id	client	chambre	arrivee	nuits
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression : $\texttt{Reservation} \bowtie_{\texttt{client}=\texttt{id}} \texttt{Client}$

Résultat :

id	client	chambre	arrivee	nuits	nom	email
1	1	504	2017-01-01	5	Jean Dupont	jean.dupont@gmail.com
2	2	107	2017-01-10	3	Alice Dupuis	alice@dupuis.name
3	3	302	2017-01-15	6	Jean Dupont	jean.dupont@ens.fr
4	2	504	2017-01-15	2	Alice Dupuis	alice@dupuis.name
5	2	107	2017-01-30	1	Alice Dupuis	alice@dupuis.name

La formule utilisée dans la jointure peut être n'importe quelle combinaison booléenne de comparaisons des valeurs d'attributs de la table de gauche avec les valeurs de la table de droite.

Note sur la jointure

- La jointure n'est pas une opération élémentaire de l'algèbre relationnelle (même si très utile)
- Elle peut être vue comme une combinaison de renommage, produit cartésien, sélection, projection
- Ainsi :

$$\begin{aligned} & \texttt{Reservation} \bowtie_{\texttt{client}=\texttt{id}} \texttt{Client} \\ \equiv\ & \Pi_{\texttt{id},\texttt{client},\texttt{chambre},\texttt{arrivee},\texttt{nuits},\texttt{nom},\texttt{email}}(\\ & \quad \sigma_{\texttt{client}=\texttt{temp}}(\texttt{Reservation} \times \rho_{\texttt{id}\to\texttt{temp}}(\texttt{Client}))) \end{aligned}$$

- Si R et S ont pour attributs $\mathcal{A}$ et $\mathcal{B}$, on note $R \bowtie S$ pour la jointure naturelle de R et de S, où la formule de jointure est $\bigwedge_{A \in \mathcal{A} \cap \mathcal{B}} A = A$.

Sémantique multi-ensembliste

En sémantique multi-ensembliste (ce qui est vraiment utilisé par les SGBDR) :

- Toutes les opérations renvoient des multi-ensembles
- En particulier, projection et union peuvent introduire des multi-ensembles même quand les relations initiales sont des ensembles

Extension : Agrégation

- Des extensions variées ont été proposées à l'algèbre relationnelle pour capturer des fonctionnalités supplémentaires
- En particulier, agrégation et regroupement [Klug, 1982, Libkin, 2003] des résultats
- Avec une syntaxe inspirée de [Libkin, 2003] :

$$\sigma_{\texttt{avg}>3}(\gamma^{\texttt{avg}}_{\texttt{chambre}}[\lambda x.\texttt{avg}(x)](\Pi_{\texttt{chambre},\texttt{nuits}}(\texttt{Reservation})))$$

calcule le nombre moyen de nuits par réservation pour chaque chambre ayant une moyenne supérieure à 3

chambre	avg
302	6
504	3,5

Plan

Calcul relationnel

- Langage logique permettant d'exprimer des requêtes
- Formule de la logique du premier ordre sans symbole de fonctions, avec comme symboles de relation les symboles du schéma relationnel
- Perspective non nommée, non typée
- On fixe :
  - Ensemble $\mathcal{X}$ de variables
  - Ensemble $\mathcal{V}$ de valeurs
  - Schéma relationnel S

Calcul relationnel : Syntaxe

- Pour toute relation $R \in S$ d'arité n, pour tout $(\alpha_1, \ldots, \alpha_n) \in (\mathcal{X} \cup \mathcal{V})^n$: $R(\alpha_1, \ldots, \alpha_n) \in \mathrm{FO}$
- Pour tout $(\phi_1, \phi_2) \in \mathrm{FO}^2$, pour tout $x \in \mathcal{X}$:
  - $\phi_1 \wedge \phi_2 \in \mathrm{FO}$
  - $\phi_1 \vee \phi_2 \in \mathrm{FO}$
  - $\neg\phi_1 \in \mathrm{FO}$
  - $\forall x\, \phi_1 \in \mathrm{FO}$
  - $\exists x\, \phi_1 \in \mathrm{FO}$

- Variables libres de $\phi \in \mathrm{FO}$: variables x apparaissant dans ϕ et non qualifiées par un $\forall x$ ou un $\exists x$
- On écrit une requête du calcul relationnel sous la forme $Q(x_1, \ldots, x_m) = \phi$ où $x_1, \ldots, x_m$ sont les variables libres de ϕ

Calcul relationnel : Sémantique

- Une requête du calcul relationnel sur le schéma S peut être vu comme une fonction prenant en entrée une base de données D sur S et renvoyant une relation
- $\mathrm{adom}(D)$: domaine actif de D, ensemble des valeurs apparaissant dans la base de données D)
- Si $Q(x_1, \ldots, x_n) = \phi$ est une requête du calcul sur S et D une base de données sur S, alors :

$$Q(D) = \{ (v_1, \ldots, v_n) \in (\mathrm{adom}(D))^n \mid D \models \phi[x_1/v_1, \ldots, x_n/v_n] \}$$

où $D \models \phi$ est défini inductivement :

- $D \models R(u_1, \ldots, u_m) \iff R(u_1, \ldots, u_m) \in D$
- $D \models \phi_1 \wedge \phi_2 \iff D \models \phi_1 \wedge D \models \phi_2$
- $D \models \phi_1 \vee \phi_2 \iff D \models \phi_1 \vee D \models \phi_2$
- $D \models \neg\phi_1 \iff D \not\models \phi_1$
- $D \models \forall x\, \phi_1 \iff \forall v \in \mathrm{adom}(D)\ D \models \phi_1[x/v]$
- $D \models \exists x\, \phi_1 \iff \exists v \in \mathrm{adom}(D)\ D \models \phi_1[x/v]$

Théorème de Codd

Théorème ([Codd, 1972])

L'algèbre relationnelle et le calcul relationnel sont équivalents :

- *pour toute requête q de l'algèbre relationnelle sur un schéma S, il existe une requête Q du calcul relationnel sur S telle que pour toute base de données D sur S, $q(D) = Q(D)$*
- *pour toute requête Q du calcul relationnel sur un schéma S, il existe une requête Q de l'algèbre relationnelle sur S telle que pour toute base de données D sur S, $q(D) = Q(D)$*

La traduction d'un formalisme vers l'autre peut se faire en temps polynomial.

Pourquoi est-ce important ?

- Permet d'utiliser un formalisme déclaratif pour exprimer des requêtes : la logique... ou SQL
- Ces requêtes sont ensuite compilées via la transformation de Codd en un formalisme algébrique
- Les requêtes algébriques sont ensuite optimisées, en utilisant les propriétés de l'algèbre relationnelle (règles de transformation, p. ex., pousser les sélections à l'intérieur des jointures, exploiter l'associativité des jointures, etc.)
- Les requêtes optimisées sont ensuite évaluées, en exploitant le fait que chaque opérateur de l'algèbre relationnel s'implémente facilement (et de plusieurs manières différentes, à choisir en fonction du coût)
- B-A-BA des SGBD relationnels, et une des raisons de leur succès !

Plan

Plan

Point fixe non-inflationniste

- On ajoute au calcul relationnel une construction de point fixe
- Supposons $\phi(T)$ formule du calcul, mentionnant les relations du schéma ainsi qu'une nouvelle relation T, et ayant pour variables libres $x_1, \ldots, x_n$
- Alors $\mu_T[\phi(T)](x_1, \ldots, x_n)$ est une formule du calcul avec point-fixe
- Sémantique : On considère le plus petit point fixe de la relation T en remplaçant T à chaque étape par l'ensemble des faits de la forme $T(x_1, \ldots, x_n)$ pour $\phi(T)(x_1, \ldots, x_n)$ satisfait (en partant de $T = \emptyset$)

Point fixe inflationniste

- On ajoute au calcul relationnel une construction de point fixe
- Supposons $\phi(T)$ formule du calcul, mentionnant les relations du schéma ainsi qu'une nouvelle relation T, et ayant pour variables libres $x_1, \ldots, x_n$
- Alors $\mu_T^+[\phi(T)](x_1, \ldots, x_n)$ est une formule du calcul avec point-fixe
- Sémantique : On considère le plus petit point fixe de la relation T en ajoutant à T à chaque étape l'ensemble des faits de la forme $T(x_1, \ldots, x_n)$ pour $\phi(T)(x_1, \ldots, x_n)$ satisfait (en partant de $T = \emptyset$)

Exemple : Clôture transitive

Non-inflationniste

$$\{(x,y) \mid \mu_C\,[G(x,y) \vee C(x,y) \vee (\exists z\, C(x,z) \wedge G(z,y))](x,y)\}$$

Inflationniste

$$\{(x,y) \mid \mu_C^+\,[G(x,y) \vee (\exists z\, C(x,z) \wedge G(z,y))](x,y)\}$$

Logique du second ordre (SO)

- Extension de la logique du premier ordre dans laquelle il est possible d'utiliser des quantificateurs sur des relations
- Par exemple, on peut exprimer en logique du second ordre qu'un graphe G est biparti :

$$\exists X \exists Y \, \forall x \forall y \, [G(x,y) \rightarrow ((X(x) \wedge Y(y)) \vee (X(y) \wedge Y(x)))] \\ \wedge \neg(\exists x \, X(x) \wedge Y(x))$$

- Peut être aussi vu comme un langage de requêtes, plus puissant que le calcul relationnel

Point fixe inflationniste et SO

- Dans certains cas, traduction naturelle de logique à point fixe vers logique du second-ordre
- En particulier, pour un point fixe inflationniste :

$$\{(x_1, \ldots, x_n) \mid \mu_T^+[\phi(T)](x_1, \ldots, x_n)\}$$

s'exprime en SO :

$$\{ (x_1, \ldots, x_n) \mid \forall T(\forall y_1 \ldots \forall y_n \, \phi(T)(y_1, \ldots, y_n) \rightarrow T(y_1, \ldots, y_n)) \rightarrow T(x_1, \ldots, x_n) \}$$

Exemple : Clôture transitive

$$\{\,(x,y)\mid$$
$$\forall C\,[\forall f \forall t\,(G(f,t) \vee (\exists z\, C(f,z) \wedge G(z,t)))] \rightarrow C(f,t))$$
$$\rightarrow C(x,y)\,\}$$

Plan

Évaluation de requêtes

- Requête Q dans un certain langage de requêtes (FO, FO+μ, FO+μ^+, SO...) – on utilisera le formalisme de la logique
- Base de données D (toujours finie !)
- Évaluation de requêtes : Calcul de $Q(D)$
- Complexité de ce problème ?
- Pour simplifier l'étude de la complexité, on considère que Q est une requête booléenne, c'est-à-dire qu'elle renvoie $\bot$ ou $\top$

Complexité en les données

Pour une certaine requête Q fixée, quelle est la complexité du calcul de $Q(D)$ en fonction de la taille de la base de données D ?

Complexité combinée

Pour un certain langage de requêtes $\mathcal{Q}$, quelle est la complexité du calcul de $Q(D)$ en fonction de la taille de la requête $Q \in \mathcal{Q}$ et de la base de données D ?

Classes de complexité

- On se restreint aux problèmes booléens (renvoyant $\bot$ ou $\top$)
- Ensemble des problèmes résolvables par une méthode de calcul à ressources contraintes :
- Par exemple :

  PTIME : Machine de Turing déterministe en temps polynomial

  NP : Machine de Turing non-déterministe en temps polynomial

  PSPACE : Machine de Turing déterministe en espace polynomial

  AC^0 : Circuit Booléen de taille polynomiale et profondeur constante
- On sait : $AC^0 \subsetneq$ PTIME $\subseteq$ NP $\subseteq$ PSPACE
- On ne sait pas si PSPACE $\subseteq$ PTIME (!)

Appartenance et difficulté pour une classe

- Un problème P appartient à une classe de complexité $\mathcal{C}$ (ou dans $\mathcal{C}$) s'il est résolvable par la méthode de calcul à ressources contraintes correspondante
- Un problème P est difficile pour la classe de complexité $\mathcal{C}$ (ou $\mathcal{C}$-difficile) s'il existe une réduction permettant de transformer n'importe quel problème $P' \in \mathcal{C}$ en une instance du problème P
- complet : dans $\mathcal{C}$ + $\mathcal{C}$-difficile
- Différentes manières de définir les réductions
- Ici, on supposera qu'il existe une fonction calculable en temps polynomial qui transforme une instance I' du problème P' en une instance I de P telle que $P(I) = P'(I')$

Complexité descriptive

- Un langage de requêtes $\mathcal{Q}$ capture une classe de complexité $\mathcal{C}$ si :
  - Pour tout $Q \in \mathcal{Q}$, le problème d'évaluation de requêtes pour Q est dans $\mathcal{C}$ (complexité en les données)
  - Pour tout problème P dans $\mathcal{C}$, il existe une requête $Q \in \mathcal{Q}$ telle que l'évaluation de Q résout exactement P (sans réduction) !
- Si $\mathcal{Q}$ capture $\mathcal{C}$ et si $\mathcal{C}$ a des problèmes complets pour $\mathcal{C}$, alors il existe $Q \in \mathcal{Q}$ telle que Q est $\mathcal{C}$-complet, mais la réciproque n'est pas vraie

Plan

Complexité dans les données

Théorème

L'évaluation de FO est PTIME en complexité dans les données.

Démonstration.

Par mise en forme prénexe et évaluation naïve. □

FO ne capture pas tout PTIME

Théorème

On ne peut pas calculer en FO qu'une relation contenant un ordre total a un nombre pair d'éléments, ou qu'un graphe est connexe.

Non prouvé, la preuve repose sur les jeux d'Ehrenfeucht–Fraïssé (voir [Libkin, 2004]).

Complexité dans les données, plus précis

Théorème

L'évaluation de FO est AC^0 en complexité dans les données.

Démonstration.

Intuition de preuve basée sur l'algèbre relationnelle. □

Complexité combinée

Théorème

L'évaluation de FO est PSPACE-complète en complexité combinée.

Démonstration.

Appartenance dans PSPACE facile. Difficulté pour PSPACE vient du problème QSAT. □

Plan

Complexité en les données

Théorème

L'évaluation de $FO{+}\mu^{+}$ est PTIME en complexité en les données.
L'évaluation de $FO{+}\mu$ est PSPACE en complexité en les données.

Démonstration.

Facile. □

Connexité, parité

Théorème

Le problème de connexité peut s'exprimer en $FO+\mu^+$.

Le problème de parité du nombre d'éléments (ou de tuples) dans une relation arbitraire ne peut pas s'exprimer en $FO+\mu$.

Démonstration.

Première partie facile, voir exemple de la clôture transitive.
Deuxième partie admise (voir [Abiteboul et al., 1995], chapitre 17). □

Parité et FO+μ^+, avec ordre

Théorème

FO+μ^+ peut calculer si une relation contenant un ordre total a un nombre pair d'éléments.

Démonstration.

Par construction explicite. □

Résultats plus généraux

Théorème

$FO{+}\mu^{+}$ exprime PTIME sur les données avec un ordre total sur les éléments du domaine.

$FO{+}\mu$ exprime PSPACE sur les données avec un ordre total sur les éléments du domaine.

Démonstration.

On simule le calcul d'une machine de Turing en temps ou espace polynomial par un calcul de point fixe, en exploitant l'ordre sur le domaine pour compter le nombre d'étapes ou l'espace utilisé. □

Complexité en les données (bis)

Théorème

L'évaluation de $FO{+}\mu^{+}$ est PTIME-complète en complexité en les données.
L'évaluation de $FO{+}\mu$ est PSPACE-complète en complexité en les données.

Démonstration.

Difficulté vient de la complexité descriptive sur les structures ordonnées. □

Au delà : Théorème de Fagin

Théorème

$\exists$SO (fragment de SO sans quantification universelle de deuxième ordre) exprime NP.

Résultat fondamental, preuve difficile.

Complexité descriptive et P≠NP

- On a une caractérisation de NP : ∃SO
- Si on avait une caractérisation de P... (indépendamment d'ordre sur les éléments du domaine !) par une classe $\mathcal{Q}$
- ... on pourrait montrer P≠NP en exhibant une requête de ∃SO non exprimable dans $\mathcal{Q}$!
- Mais on en est loin...

Plan

Modèle de données

- Modèle de données relationnel : données décomposées en relations, avec attributs étiquetés...

Modèle de données

- Modèle de données relationnel : données décomposées en relations, avec attributs étiquetés...

nom	poste	ville
John	Directeur	New York
Paul	Gardien	New York
Dave	Analyste	Paris
Ellen	Agent	Berlin
Magdalen	Agent double	Paris
Nancy	DRH	Paris
Susan	Analyste	Berlin

Modèle de données

- Modèle de données relationnel : données décomposées en relations, avec attributs étiquetés...
- ... et une annotation de provenance supplémentaire pour chaque tuple (la voir comme un identifiant de tuple)

nom	poste	ville	prov
John	Directeur	New York	t_1
Paul	Gardien	New York	t_2
Dave	Analyste	Paris	t_3
Ellen	Agent	Berlin	t_4
Magdalen	Agent double	Paris	t_5
Nancy	DRH	Paris	t_6
Susan	Analyste	Berlin	t_7

Provenance booléenne [Imielinski and Jr., 1984]

- $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ ensemble fini d'événements booléens
- annotation de provenance : fonction booléenne sur $\mathcal{X}$, c.-à-d., une fonction de la forme : $(\mathcal{X} \to \{\bot, \top\}) \to \{\bot, \top\}$
- Interprétation : sémantique de mondes possibles
  - chaque valuation $\nu : \mathcal{X} \to \{\bot, \top\}$ désigne un monde possible de la base de données
  - la provenance d'un tuple ν évalue à $\bot$ ou $\top$ suivant si ce tuple existe dans ce monde possible
  - par exemple, si chaque tuple de la base est annoté avec la fonction indicatrice d'un événement booléen distinct, l'ensemble des mondes possibles est l'ensemble de toutes les sous-instances

Exemples de mondes possibles

nom	poste	ville	prov
John	Directeur	New York	t_1
Paul	Gardien	New York	t_2
Dave	Analyste	Paris	t_3
Ellen	Agent	Berlin	t_4
Magdalen	Agent double	Paris	t_5
Nancy	DRH	Paris	t_6
Susan	Analyste	Berlin	t_7

$$\nu : \begin{array}{ccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \\ \top & \top & \top & \top & \top & \top & \top \end{array}$$

Exemples de mondes possibles

nom	poste	ville	prov
John	Directeur	New York	t_1
Dave	Analyste	Paris	t_3
Magdalen	Agent double	Paris	t_5
Susan	Analyste	Berlin	t_7

$$\nu : \begin{array}{cccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \\ \top & \bot & \top & \bot & \top & \bot & \top \end{array}$$

Provenance booléenne des résultats de requêtes

- $\nu(D)$: la sous-instance de D où tous les tuples dont l'annotation de provenance évalue à $\bot$ par ν sont enlevés
- La provenance booléenne $\text{prov}_{q,D}(t)$ d'un tuple $t \in q(D)$ est la fonction :

$$\nu \mapsto \begin{cases} \top \text{ si } t \in q(\nu(D)) \\ \bot \text{ sinon} \end{cases}$$

Exemple (Quelles villes sont dans la table Personnel ?)

nom	poste	ville	prov
John	Directeur	New York	t_1
Paul	Gardien	New York	t_2
Dave	Analyste	Paris	t_3
Ellen	Agent	Berlin	t_4
Magdalen	Agent double	Paris	t_5
Nancy	DRH	Paris	t_6
Susan	Analyste	Berlin	t_7

ville	prov
New York	$t_1 \vee t_2$
Paris	$t_3 \vee t_5 \vee t_6$
Berlin	$t_4 \vee t_7$

Semi-anneau commutatif $(K, \mathbb{0}, \mathbb{1}, \oplus, \otimes)$

- Ensemble K avec éléments distingués $\mathbb{0}$, $\mathbb{1}$
- $\oplus$ opérateur associatif, commutatif, avec neutre $\mathbb{0}_K$:
  - $a \oplus (b \oplus c) = (a \oplus b) \oplus c$
  - $a \oplus b = b \oplus a$
  - $a \oplus \mathbb{0} = \mathbb{0} \oplus a = a$
- $\otimes$ opérateur associatif, commutatif, avec neutre $\mathbb{1}_K$:
  - $a \otimes (b \otimes c) = (a \otimes b) \otimes c$
  - $a \otimes b = b \otimes a$
  - $a \otimes \mathbb{1} = \mathbb{1} \otimes a = a$
- $\otimes$ distribue sur $\oplus$:

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$$

- $\mathbb{0}$ est annulateur pour $\otimes$:

$$a \otimes \mathbb{0} = \mathbb{0} \otimes a = \mathbb{0}$$

Exemples de semi-anneaux

- $(\mathbb{N}, 0, 1, +, \times)$: semi-anneau de comptage
- $(\{\bot, \top\}, \bot, \top, \vee, \wedge)$: semi-anneau booléen
- $(\{\mathit{unclassified}, \mathit{restricted}, \mathit{confidential}, \mathit{secret}, \mathit{top\ secret}\}, \mathit{top\ secret}, \mathit{unclassified}, \min, \max)$: semi-anneau de sécurité
- $(\mathbb{N} \cup \{\infty\}, \infty, 0, \min, +)$: semi-anneau tropical
- $(\{\text{fonctions booléennes sur } \mathcal{X}\}, \bot, \top, \vee, \wedge)$: semi-anneau des fonctions booléennes sur $\mathcal{X}$
- $(\mathcal{P}(\mathcal{P}(\mathcal{X})), \emptyset, \{\emptyset\}, \cup, \Cup)$: Semi-anneau Why sur $\mathcal{X}$ $(A \Cup B = \{a \cup b \mid a \in A, b \in B\})$

Provenance de semi-anneau [Green et al., 2007]

- On fixe un semi-anneau $(K, \mathbb{0}, \mathbb{1}, \oplus, \otimes)$
- On suppose que les annotations de provenance sont dans K
- On considère une requête q de l'algèbre relationnelle positive (sélection, projection, renommage, produit cartésien, union ; jointures simulables avec renommage, produit cartésien, sélection, projection)
- On définit la sémantique de la provenance d'un tuple $t \in q(D)$ récursivement sur la structure de q

Sélection, renommage

Les annotations de provenance des tuples sélectionnés sont inchangées

Exemple $(\rho_{\text{nom}\to\text{n}}(\sigma_{\text{ville}=\text{“New York”}}(R)))$

nom	poste	ville	prov
John	Directeur	New York	t_1
Paul	Gardien	New York	t_2
Dave	Analyste	Paris	t_3
Ellen	Agent	Berlin	t_4
Magdalen	Agent double	Paris	t_5
Nancy	DRH	Paris	t_6
Susan	Analyste	Berlin	t_7

n	poste	ville	prov
John	Directeur	New York	t_1
Paul	Gardien	New York	t_2

Projection

Les annotations de provenance des tuples identiques, fusionnés, sont $\oplus$-ées

Exemple ($\pi_{\text{ville}}(R)$)

nom	poste	ville	prov
John	Directeur	New York	t_1
Paul	Gardien	New York	t_2
Dave	Analyste	Paris	t_3
Ellen	Agent	Berlin	t_4
Magdalen	Agent double	Paris	t_5
Nancy	DRH	Paris	t_6
Susan	Analyste	Berlin	t_7

ville	prov
New York	$t_1 \oplus t_2$
Paris	$t_3 \oplus t_5 \oplus t_6$
Berlin	$t_4 \oplus t_7$

Union

Les annotations de provenance des tuples identiques, fusionnés, sont $\oplus$-ées

Exemple

$\pi_{\text{ville}}(\sigma_{\textit{ends-with}(\text{poste},\text{“agent”})}(R)) \cup \pi_{\text{ville}}(\sigma_{\text{poste}=\text{“Analyste”}}(R))$

nom	poste	ville	prov
John	Directeur	New York	t_1
Paul	Gardien	New York	t_2
Dave	Analyste	Paris	t_3
Ellen	Agent	Berlin	t_4
Magdalen	Agent double	Paris	t_5
Nancy	DRH	Paris	t_6
Susan	Analyste	Berlin	t_7

ville	prov
Paris	$t_3 \oplus t_5$
Berlin	$t_4 \oplus t_7$

Produit cartésien

Les annotations de provenance des tuples combinés sont $\otimes$-ées

Exemple

$\pi_{\text{ville}}(\sigma_{\textit{ends-with}(\text{poste},\text{“agent”})}(R)) \bowtie \pi_{\text{ville}}(\sigma_{\text{poste}=\text{“Analyste”}}(R))$

nom	poste	ville	prov
John	Directeur	New York	t_1
Paul	Gardien	New York	t_2
Dave	Analyste	Paris	t_3
Ellen	Agent	Berlin	t_4
Magdalen	Agent double	Paris	t_5
Nancy	DRH	Paris	t_6
Susan	Analyste	Berlin	t_7

ville	prov
Paris	$t_3 \otimes t_5$
Berlin	$t_4 \otimes t_7$

Que peut-on faire avec ça ?

semi-anneau de comptage : compter le nombre de fois qu'un tuple peut être dérivé, sémantique multi-ensemble

semi-anneau booléen : détermine si un tuple existe quand une sous-base de données est sélectionnée

semi-anneau de sécurité : détermine le niveau de confidentialité minimale requis pour savoir qu'un tuple est dans le résultat

semi-anneau tropical : coût minimal de dérivation d'un tuple (penser aux plus cours chemins dans un graphe)

Fonctions booléennes : provenance booléenne, comme définie précédemment

Semi-anneau Why : Why-provenance [Buneman et al., 2001], ensemble des combinaisons de tuples nécessaires à l'existence d'un tuple

Exemple de Why-provenance

$$\pi_{\text{ville}}(\sigma_{\text{nom}<\text{nom2}}(\pi_{\text{nom,ville}}(R) \bowtie \rho_{\text{nom}\to\text{nom2}}(\pi_{\text{nom,ville}}(R))))$$

nom	poste	ville	prov
John	Directeur	New York	t_1
Paul	Gardien	New York	t_2
Dave	Analyste	Paris	t_3
Ellen	Agent	Berlin	t_4
Magdalen	Agent double	Paris	t_5
Nancy	DRH	Paris	t_6
Susan	Analyste	Berlin	t_7

ville	prov
New York	$\{\{t_1, t_2\}\}$
Paris	$\{\{t_3, t_5\}, \{t_3, t_6\}, \{t_5, t_6\}\}$
Berlin	$\{\{t_4, t_7\}\}$

Remarques [Green et al., 2007]

- Le calcul de la provenance a un surcoût polynomial en temps
- Deux requêtes équivalentes peuvent avoir deux annotations différentes de provenance sur la même base de données, dans certains semi-anneaux (par exemple, dans le semi-anneau Why)

Plan

BD et informatique théorique

- Riches connexions entre la gestion de données et l'informatique théorique : les BD, ce n'est pas (seulement) de l'ingénierie !
- Plein d'autres choses encore :

  BD et automates : évaluation en $O(n)$ de requêtes XPath sur des arbres XML grâce aux automates d'arbres [Neven, 2002]

  BD et complexité paramétrée : Calcul efficace de probabilité d'une requête sur une BD probabiliste si le graphe de la BD a une largeur d'arbre bornée [Amarilli et al., 2015]

  BD et hypergraphes : Algorithme de jointure optimal dans le pire des cas basé sur la couverture fractionnaire par arêtes de l'hypergraphe de la BD [Ngo et al., 2018]

Références

- Généralités sur la gestion de données [Benedikt and Senellart, 2012, Abiteboul, 2012]
- Cours sur le programme de prépa en bases de données [Abiteboul et al., 2014]
- Référence sur la théorie des bases de données classique : [Abiteboul et al., 1995]
- Référence sur la théorie des modèles finis : [Libkin, 2004]
- Introduction à la provenance : [Senellart, 2017]

Bibliographie I

Serge Abiteboul. *Sciences des données : de la logique du premier ordre à la Toile*. Collège de France, 2012. `http://books.openedition.org/cdf/529`.

Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

Serge Abiteboul, Benjamin Nguyen, and Yannick Le Bras. Introduction aux bases de données relationnelles. `http://abiteboul.com/Lili/bdrelationnelles.pdf`, 2014.

Antoine Amarilli, Pierre Bourhis, and Pierre Senellart. Provenance circuits for trees and treelike instances. In *Proc. ICALP*, pages 56–68, Kyoto, Japan, July 2015.

Michael Benedikt and Pierre Senellart. Databases. In Edward K. Blum and Alfred V. Aho, editors, *Computer Science. The Hardware, Software and Heart of It*, pages 169–229. Springer-Verlag, 2012.

Bibliographie II

Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Why and where : A characterization of data provenance. In *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings.*, 2001.

Edgar F. Codd. Relational completeness of data base sublanguages. In *Data Base Systems. Courant Computer Science Symposium*, 1972.

Todd J Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *PODS*, 2007.

Tomasz Imielinski and Witold Lipski Jr. Incomplete information in relational databases. *J. ACM*, 31(4), 1984.

Anthony C. Klug. Equivalence of relational algebra and relational calculus query languages having aggregate functions. *J. ACM*, 29(3) :699–717, 1982.

Bibliographie III

Leonid Libkin. Expressive power of SQL. *Theor. Comput. Sci.*, 296(3) :379–404, 2003.

Leonid Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004. doi : 10.1007/978-3-662-07003-1. URL `http://dx.doi.org/10.1007/978-3-662-07003-1`.

Frank Neven. Automata theory for XML researchers. *SIGMOD Record*, 31(3) :39–46, 2002. doi : 10.1145/601858.601869. URL `http://doi.acm.org/10.1145/601858.601869`.

Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. Worst-case optimal join algorithms. *J. ACM*, 65(3) : 16 :1–16 :40, 2018. doi : 10.1145/3180143. URL `http://doi.acm.org/10.1145/3180143`.

Pierre Senellart. Provenance and probabilities in relational databases: From theory to practice. *SIGMOD Record*, 46(4), December 2017.