Université de Bordeaux
M1 d'Informatique, 2016-2017
Cours de Bases de Données Avancées

Feuille 1
Modèle Relationnel et Requêtes Conjonctives

Le but de cette feuille est d'introduire le modèle de bases de données le plus classique : les *bases de données relationnelles*. Elle introduit également un premier langage de requêtes : les *requêtes conjonctives* (CQ).

1 Bases de Données Relationnelles

Comme le nom l'indique les données sont rangées dans des *relations* qui permettent de lier ces données entres elles. On distingue deux notions bien différentes :

- le *schéma de base de données*, **R**. Il fixe les noms de relations qu'on peut utiliser et comment y ranger les données.
  Attention : Le schéma **ne contient aucune données**, il ne fait que spécifier la façon dont on va ensuite ranger les données.
- les *instances d'un schéma de base de données*. Une instance est définie par rapport à un schéma **R**. Elle contient des données rangées selon la spécification donnée par **R**.

1.1 Schéma de Base de Données

Un schéma de bases de données est composé de trois éléments :

- **rel** : un ensemble fini de noms de *relations*.
- **att** : un ensemble fini d'*attributs*.
- Une fonction *sort* qui associe une liste (ordonnée) d'attributs à chaque relation de **rel**.

Pour chaque relation $R \in \mathbf{rel}$, on appellera *arité* de R le nombre d'attributs qui lui sont associés par *sort*.

Exemple 1 *On présente le schéma* **COURS** *:*

$\mathbf{rel} = \{Personnel, Locaux, Informations\}$
$\mathbf{att} = \{Salle, Horaire, Bâtiment, Intitulé, Enseignant, Étudiant\}$

$$\begin{array}{lcl} sort(Personnel) & = & Intitulé, Enseignant, Étudiant \\ sort(Locaux) & = & Salle, Bâtiment \\ sort(Informations) & = & Salle, Horaire, Intitulé \end{array}$$

1.2 Instance d'un Schéma de Base de Données

On fixe un ensemble infini de *données* qu'on appelle le *domaine*, **dom**. Typiquement, dans nos exemples, le domaine sera simplement l'ensemble des chaînes de caractères. Une instance dépend du choix du domaine **dom** (puisque elle est constituée d'éléments de **dom**).

Prenons un schéma **R**. Un *fait* pour **R** (sur **dom**) est noté $R(d_1, \ldots, d_n)$ où R est une relation de R d'arité n et $(d_1, \ldots, d_n)$ est un tuple de n données. Intuitivement d_1 est associé au premier attribut de R, d_2 au second, etc...

Remarque 1 *Comme on le voit ci-dessus, on sera souvent amené à manipuler des tuples. Pour simplifier les notations, on abrégera souvent un tuple* $(x_1, \ldots, x_n)$ *par* $\bar{x}$.

On appliquera aussi souvent des fonctions à tous les éléments d'un tuples. Dans ce cas, on notera $f(\bar{x})$ *pour* $(f(x_1), \ldots, f(x_n))$.

Une instance **I**, de **R** (sur le domaine **dom**) est un *ensemble fini de faits* (sur **dom**). Par commodité on représentera souvent les instances sous forme de tableaux (voir l'exemple ci-dessous).

Exemple 2 *Instance du schéma* **COURS** *de l'Exemple 1*

Personnel

Intitulé	*Enseignant*	*Etudiant*
BDA	*T. Place*	*Bob*
BDA	*T. Place*	*Roger*
...	...	...
Algo	*J. Prof*	*Bob*
Algo	*J. Prof*	*Alice*
...	...	...

Locaux

Salle	*Bâtiment*
1	*A*
2	*A*
...	...
560	*Z*
561	*Z*
...	...

Informations

Salle	*Horaire*	*Intitulé*
42	*Jeudi 8h*	*BDDA*
...	...	...
42bis	*Lundi 14h*	*Logique*
...	...	...

En particulier, $Personnel(Algo, J.Prof, Alice)$ *et* $Locaux(1, A)$ *sont des exemples de faits dans l'instance.*

Attention à la différence entre schéma et instance. Un schéma est une spécification décrivant une façon de ranger des données. Une instance de ce schéma est un ensemble de données rangées selon cette spécification. En particulier, il existe une *infinité d'instances possibles* d'un même schéma.

Enfin, notons qu'il est possible et naturel de comparer deux instances d'un même schéma par l'*inclusion*. Si $\mathbf{I}_1, \mathbf{I}_2$ sont deux instances d'un même schéma **R**, on note

$$\mathbf{I}_1 \subseteq \mathbf{I}_2$$

quand tous les faits de $\mathbf{I}_1$ sont également des faits de $\mathbf{I}_2$. Intuitivement, celà signifie que $\mathbf{I}_2$ contient *plus d'information* que $\mathbf{I}_2$.

2 Requêtes Conjonctives

On va maintenant présenter un premier langage de requêtes qui permet d'extraire des informations d'une instance de base de données : les *requêtes conjonctives* (nous noterons ce langage CQ). Bien que très simple et peu expressif, il a l'avantage de posséder de bonnes propriétés algorithmiques. Le choix d'un langage de requêtes est un compromis entre trois ambitions (potentiellement contradictoires) :

1. Expressivité : Que peut-on (et que ne peut-on pas) exprimer avec le langage de requête ?
2. Évaluation : Quelle est la complexité de l'évaluation d'une requête du langage ?
3. Analyse Statique : Quelle est la complexité de l'analyse d'une ou plusieurs requêtes ?

Commençons par décrire plus en détail ce qu'on attend pour l'évaluation et l'analyse statique des requêtes.

2.1 Évaluation

On cherche un langage de requêtes pour lequel les requêtes sont faciles à évaluer. On distinguera deux types de complexités pour les requêtes :

1. Complexité Combinée : Étant donné une requête q et une instance $\mathbf{I}$, quelle est la complexité (en fonction de q et $\mathbf{I}$) requise pour évaluer $q(\mathbf{I})$.
2. Complexité dans les Données. Supposons que la requête q est fixée. Étant donné une instance $\mathbf{I}$, quelle est la complexité (en fonction de $\mathbf{I}$) requise pour évaluer $q(\mathbf{I})$

2.2 Analyse Statique

Le but de l'analyse statique est d'optimiser une requête fixée q pour l'évaluation. Essentiellement cela revient à étudier trois problèmes de décision. Idéalement, on souhaite avoir des algorithmes qui traitent ces problèmes. Le premier concerne le traitement algorithmique d'une requête.

Problème 1 *Satisfiabilité :*
INPUT : *Une requête q.*
OUTPUT : *Existe-t'il une instance $\mathbf{I}$ pour laquelle $q(\mathbf{I})$ est non-vide ?*

Les deux autres problèmes concernent la comparaison de requêtes entre elles. Si q et r sont des requêtes, on dit que q est incluse dans r (noté $q \subseteq r$) si pour toute instance $\mathbf{I}$, $q(\mathbf{I}) \subseteq r(\mathbf{I})$. De même on dit que q et r sont équivalentes (noté $q \sim r$) si pour toute instance $\mathbf{I}$, $q(\mathbf{I}) = r(\mathbf{I})$. En particulier $q \sim r$ si et seulement si $q \subseteq r$ er $q \supseteq r$.

Problème 2 *Inclusion :*
INPUT : *Deux requêtes q et r.*
OUTPUT : *Est-ce que $q \subseteq r$?*

Problème 3 *Équivalence :*
INPUT : *Deux requêtes q et r.*
OUTPUT : *Est-ce que $q \sim r$?*

2.3 Définition

On peut maintenant définir les requêtes conjonctives. On notera cette classe de requêtes CQ. On procède en deux temps. On commence par définir leur syntaxe (comment s'écrit une requête conjonctive), avant de définir leur sémantique (étant donné une requête quelle est son évaluation sur une instance donnée).

Syntaxe. Pour la définition, on suppose que le domaine **dom** et le schéma de base de données **R** sont tous les deux fixés. On fixe également un ensemble infini de variables, **var** qu'on va utiliser pour écrire nos requêtes. Une requête conjonctive q (pour le schéma **R**) s'écrit de la façon suivante :

$$q(\bar{z}) \leftarrow R_1(\bar{y_1}), \ldots, R_n(\bar{y_n})$$

Pour tout i, R_i doit être une relation de **R** et $\bar{y_i}$ doit être un tuple constitué de variables (dans **var**) et d'éléments du domaine (dans **dom**). De plus, la longueur de $\bar{y_i}$ doit être égale à l'arité de R_i que nous noterons n_i. Autrement dit, $\bar{y_i} \in (\mathbf{var} \cup \mathbf{dom})^{n_i}$. Enfin $\bar{z}$ est également un tuple constitué de variables (dans **var**) et d'éléments du domaine (dans **dom**). La longueur de $\bar{z}$ est libre. On appelle cette longueur ℓ l'*arité de la requête* q. Enfin toutes les variables de $\bar{z}$ **doivent apparaître au moins une fois** dans un des tuples $\bar{y_1}, \ldots, \bar{y_n}$.

Les variables de $\bar{z}$ sont les *variables libres* de la requête. $q(\bar{z})$ est appelée la *tête* de la requête et $R_1(\bar{y_1}), \ldots, R_n(\bar{y_n})$ le *corps* de la requête.

Exemple 3 *On reprend le schéma* **COURS**. *Voici un exemple de requête conjonctive qui n'utilise qu'une seule relation.*

$$q(u) \leftarrow Personnel('Algorithmique', u, y)$$

Sémantique. Maintenant que nous avons défini ce qu'est une requête conjonctive, nous pouvons donner une sémantique à chaque requête : étant donnée une requête q et une instance **I** fixées, on définit l'évaluation $q(\mathbf{I})$ de q sur **I**.

On appelle *affectation* une fonction $f : (\mathbf{var} \cup \mathbf{dom}) \to \mathbf{dom}$ telle que pour tout $c \in \mathbf{dom}$, $f(c) = c$. Autrement dit, une affectation affecte une donnée dans **dom** à chaque variable dans **var** et ne modifie pas les éléments de **dom**.

Soit $q(\bar{z}) \leftarrow R_1(\bar{y_1}), \ldots, R_n(\bar{y_n})$ une requête conjonctive et **I** une instance. Étant donné une affectation f, on dit que $f, \mathbf{I}$ *satisfait* q (noté $f, \mathbf{I} \models q$) si et seulement si pour tout i, $R_i(f(\bar{y_i})) \in \mathbf{I}$. On peut maintenant définir l'évaluation de q sur **I** :

$$q(\mathbf{I}) = \{f(\bar{z}) \mid f, \mathbf{I} \models q\}$$

Exercice 1

On considère le schéma de base de données **BOUTIQUE** contenant les relations $\{Ventes, Type, Clients, Fabriquants\}$ telles que :

- $sort(Ventes) = VNom, FNom, CNom.$
- $sort(Type) = VNom, Type.$
- $sort(Clients) = CNom, CAddr.$
- $sort(Fournisseurs) = FNom, FAddr.$

On donne ci-dessous une instance de **BOUTIQUE** :

Ventes	VNom	FNom	CNom
	Papier	Grossiste	Univ Bordeaux
	PC	Dell	Univ Bordeaux
	Papier	Auchan	Durand
	Agraphes	Grossiste	Dupont

Type	VNom	Type
	PC	Ordinateur
	Papier	Fournitures
	Agraphes	Fournitures

Clients	CNom	CAddr
	Univ Bordeaux	Bordeaux
	Dupont	Paris
	Durand	Bordeaux

Fournisseurs	FNom	FAddr
	Grossiste	Bordeaux
	Auchan	Bordeaux
	Dell	Round Rock

Donner l'évaluation des requêtes suivantes :

1. $q_1(x) \leftarrow Ventes(z, y, x), Clients(x, Bordeaux)$
2. $q_2(x, y) \leftarrow Ventes(y, x, Univ), Fournisseurs(x, Bordeaux), Type(y, Fournitures)$
3. $q_3(x, y) \leftarrow Ventes(y, x, Univ), Fournisseurs(x, Bordeaux), Type(z, Fournitures)$
4. $q_4(x, y) \leftarrow Clients(x, Bordeaux), Clients(y, Bordeaux), Ventes(a1, b1, x), Ventes(a2, b2, y), Fournisseurs(b1, z), Fournisseurs(b2, z)$

Exercice 2
On reprend le schéma **COURS** vu précedemment. Donner des requêtes conjonctives pour les requêtes suivantes :

1. Qui enseigne le cours d'algorithmique ?
2. Bob a-t'il cours dans la salle 42 ?
3. Lister tous les enseignants faisant cours dans le bâtiment C.
4. Lister les paires (enseignant,élève) tels que l'élève suit un cours de l'enseignant.
5. Lister les paires d'élèves qui vont aux mêmes cours.
6. Quels cours ayant J. Prof pour enseignant ont lieu en salle 42 ?

Exercice 3
Soit **R** un schéma de base de données, $q \in$ CQ et **I** une instance. Donnez une borne supérieure en fonction de **I** et de l'arité de q sur la taille de $q(\mathbf{I})$.

2.4 Satisfiabilité

Proposition 1 *Le problème de satisfiabilité d'une requête conjonctive est décidable.*

Exercice 4
Prouver la Proposition 1. On demande donc de donner un algorithme qui teste la satisfiabilité des requêtes conjonctives.

2.5 Monotonicité

Une propriété importante des requêtes conjonctives est leur monotonicité :

Proposition 2 *Les requêtes conjonctives sont monotones. Autrement dit, si* $q \in$ CQ*, pour toutes instances* **I,J** *:*

$$\mathbf{I} \subseteq \mathbf{J} \Rightarrow q(\mathbf{I}) \subseteq q(\mathbf{J})$$

Exercice 5
Prouver la Proposition 2.

La monotonicité est très utile pour montrer qu'une requête n'est pas conjonctive. Si une requête q n'est pas monotone (donc si on peut trouver **I,J** telles que $\mathbf{I} \subseteq \mathbf{J}$ et $q(\mathbf{I}) \not\subseteq q(\mathbf{J})$), alors elle n'est pas exprimable par une requête conjonctive.

Exercice 6
On considère le schéma de base de données **CINEMA** contenant les relations $\{Films, Lieux, Guide\}$ tels que :

- $sort(Films)$ = Titre, Réalisateur, Acteur.
- $sort(Lieux)$ = Salle,Adresse,Téléphone.
- $sort(Guide)$ = Salle,Titre,Heure.

Pour chacune des requêtes suivantes, dites si elle est exprimable en calcul conjonctif. Si la réponse est non, donnez une preuve (on pourra utiliser la monotonicité des requêtes conjonctives). Si la réponse est oui donnez la formule du calcul conjonctif correspondante.

1. Qui a réalisé 'Brazil' ?
2. Dans quel cinémas peut-on voir 'Brazil' ?
3. Quels sont l'adresse et le numéro de téléphone du 'Ciné B' ?
4. Lister les noms et adresses des cinémas passant un film de 'Terry Gilliam'.
5. Y a t'il un film de 'Terry Gilliam' qui passe a 'Bordeaux' ?
6. Lister les paires de personnes telles que la première a dirigé la seconde et vice versa.
7. Lister les réalisateurs qui ont joué dans un film qu'ils ont dirigé.
8. Lister les paires d'acteurs qui ont joué dans le même film.
9. Pour toute entrée répondre ('Apocalypse Now','Coppola').
10. Lister tous les acteurs et réalisateurs d'Apocalypse Now'.
11. Lister les films réalisés par 'Spielberg' mais sans 'Harrison Ford'.
12. Lister les réalisateurs ayant réalisé un nombre pair de films.
13. Lister les films qui ne sont joués que dans une seule salle.

3 Exercices supplémentaires

Exercice 7

On ajoute l'égalité '=' au calcul conjonctif (c'est à dire que la conjonction peut maintenant contenir des égalités entre variables et constantes).

1. Soit $\mathbf{R}$ un schéma de bases de données. La sémantique $q(\mathbf{I})$ pour une requête q est elle bien définie pour toute instance $\mathbf{I}$.
2. Donner une condition syntaxique simple sur la requête q qui pallie à ce problème.
3. Donner une condition sémantique simple qui pallier à ce problème.

Exercice 8

Un *atome d'inégalité* est une expression de la forme $x \neq a$ ou $x \neq y$ pour x, y des variables et a une constante. On suppose que le domaine **dom** est muni d'un ordre total $\leq$, un *atome de comparaison* est une expression $x\theta y$ ou $x\theta a$ pour $\theta = \leq, \geq, <$ ou $>$, x, y des variables et a une constante.

1. Montrez que le calcul conjonctif avec inégalité et égalité est strictement plus expressif que le calcul conjonctif avec égalité.
2. Comparez les pouvoirs d'expression des langages de requête suivants :
   — Calcul conjonctif avec égalité.
   — Calcul conjonctif avec égalité et inégalité.
   — Calcul conjonctif avec égalité et comparaison.
   — Calcul conjonctif avec comparaison.
   — Calcul conjonctif avec égalité, inégalité et comparaison.

Exercice 9

Parmi les requêtes suivantes, donnez les inclusions et les équivalences (on ne demande pas de justifier pour l'instant) :

— $q_1(x) \leftarrow R(x, x)$
— $q_2(x) \leftarrow R(x, y)$
— $q_3(x) \leftarrow R(x, y), R(y, x)$
— $q_4(x) \leftarrow R(x, z), R(z, x)$
— $q_5(x) \leftarrow R(x, z), Q(y, y)$
— $q_6(x) \leftarrow Q(y, y)$
— $q_6(x) \leftarrow Q(z, x), Q(x, y)$
— $q_7(x) \leftarrow Q(z, x), Q(x, y), Q(a, y), Q(z, b), Q(c, x)$