Langages de requêtes en bases de données: tour d'horizon des différents paradigmes

Emmanuel Coquery

17 mars 2016

Requêtes

Obtenir des informations à partir d'un "paquet" de données

Requêtes servent à spécifier:

- Quelles sont les données à utiliser
  - Quelles sont les données intéressantes
  - Quelle donnée correspond à quelle autre
- Comment combiner ces données
  - Quels calculs effectuer sur ces données

Quel langage pour interroger quelles données ?

Modèles de données

Manière dont est organisée l'information

- Données tabulaires (dont modèle relationnel):
  - Ensemble de tableaux
  - attributs en colonne
  - ligne = un "objet" / une association entre "objets"
  - contenu d'une case: type *atomique*
- Données en arbre (XML, JSON)
  - Données atomiques plutôt sur les feuilles
  - Noeud internes nommés $\rightarrow$ organise les données
- Données en graphe (RDF)
  - Arêtes et noeuds étiquetés
  - Données atomiques: en général sur les noeuds

Paradigmes et modèles de données

- Logique
  - Modèles: tabulaire, graphes (+arbres)
- Navigationel
  - Modèles: arbres, graphes
  - Souvent intégrés à d'autres paradigmes (logique, fonctionnel)
- Algébrique
  - Tous les modèles
  - Utilisés pour modéliser les plan d'exécution

Langages déclaratifs

Opérations fondamentales

- Sélection: filtrage des données
  - condition booléenne sur les valeurs
- Jointure
  - Mise en correspondance des données
- Projection
  - Suppression d'une partie de la structure (e.g. attributs) non intéressante
  - Calculs sur les valeurs atomiques

Plan

1. Introduction
2. Langages logiques
3. Langages navigationnels
4. Langages algébriques
5. Conclusion

1 Introduction

2 Langages logiques
- SQL et Calcul relationnel
- Datalog
- SPARQL

3 Langages navigationnels

4 Langages algébriques

5 Conclusion

Relation exemple

Empno	Lastname	Workdept	Job	Educlevel	Sex	Sal	Bonus	Comm	Mgrno
10	SPEN	C01	FINANCE	18	F	52750	500	4220	20
20	THOMP	-	MANAGER	18	M	41250	800	3300	-
30	KWAN	-	FINANCE	20	F	38250	500	3060	10
50	GEYER	-	MANAGER	16	M	40175	800	3214	20
60	STERN	D21	SALE	14	M	32250	500	2580	30
70	PULASKI	D21	SALE	16	F	36170	700	2893	100
90	HENDER	D21	SALE	17	F	29750	500	2380	10
100	SPEN	C01	FINANCE	18	M	26150	800	2092	20

Relation `Emp`

Exemple de requête SQL

```
SELECT e1.Lastname, e1.Sex, e1.Sal
FROM Emp e1, Emp e2
WHERE e1.Lastname = e2.Lastname
  AND e1.Workdept = e2.Workdept
  AND e1.Empno <> e2.Empno
```

Lastname	Sex	Sal
SPEN	F	52750
SPEN	M	26150

Calcul relationnel

On suppose fixé:

- Ensemble dénombrable de constantes: Domaine $\mathcal{D}$
- Ensemble fini d'attributs: Univers $\mathcal{U}$
- Ensemble fini de symboles de relation $\mathcal{R}$

Tuples, relations, schémas, instances

Definition (Tuple)

Soit $X \subseteq \mathcal{U}$. Un *tuple* $\mathtt{t}$ sur X est une fonction $X \to \mathcal{D}$.

Definition (Relation)

Une *relation* $\mathtt{r}$ sur X est un ensemble de tuples sur X.

Definition (Schema)

Ensemble d'attributs $X \subseteq \mathcal{U}$ associé à un symbole de relation / une relation / un tuple.

Definition (Instance)

Instance de $R \in \mathcal{R}$: relation $\mathtt{r}$ sur X
Instance d d'une base de donnée: fonction associant une instance à chaque symbole de relation $R \in \mathcal{R}$

SQL → Tuple Relationnal Calculus

```
SELECT e1.Lastname, e1.Sex, e1.Sal
FROM Emp e1, Emp e2
WHERE e1.Lastname = e2.Lastname
  AND e1.Workdept = e2.Workdept
  AND e1.Empno <> e2.Empno
```

$$\{t \mid \exists t_1 \exists t_2$$
$$t.Lastname = t_1.Lastname \wedge t.Sex = t_1.Sex \wedge t.Sal = t_1.Sal$$
$$Emp(t_1) \wedge Emp(t_2)$$
$$\wedge t_1.Lastname = t_2.Lastname \wedge t_1.Workdept = t2.Workdept$$
$$\wedge t_1.Empno \neq t_2.Empno\}$$

SQL $\rightarrow$ Tuple Relationnal Calculus

```
SELECT e1.Lastname, e1.Sex, e1.Sal
FROM Emp e1, Emp e2
WHERE e1.Lastname = e2.Lastname
  AND e1.Workdept = e2.Workdept
  AND e1.Empno <> e2.Empno
```

$$\{t \mid \exists t_1 \exists t_2$$
$$t.Lastname = t_1.Lastname \wedge t.Sex = t_1.Sex \wedge t.Sal = t_1.Sal$$
$$Emp(t_1) \wedge Emp(t_2)$$
$$\wedge t_1.Lastname = t_2.Lastname \wedge t_1.Workdept = t2.Workdept$$
$$\wedge t_1.Empno \neq t_2.Empno\}$$

SQL → Tuple Relationnal Calculus

```
SELECT e1.Lastname, e1.Sex, e1.Sal
FROM Emp e1, Emp e2
WHERE e1.Lastname = e2.Lastname
  AND e1.Workdept = e2.Workdept
  AND e1.Empno <> e2.Empno
```

$$\{t \mid \exists t_1 \exists t_2$$
$$t.Lastname = t_1.Lastname \wedge t.Sex = t_1.Sex \wedge t.Sal = t_1.Sal$$
$$Emp(t_1) \wedge Emp(t_2)$$
$$\wedge t_1.Lastname = t_2.Lastname \wedge t_1.Workdept = t2.Workdept$$
$$\wedge t_1.Empno \neq t_2.Empno\}$$

Formules du TRC

Definition (Syntaxe)

$$\psi ::= \mathtt{R}(t) \mid t_1.\mathtt{A} \square t_2.\mathtt{B} \mid t.\mathtt{A} \square \mathtt{c} \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid \exists t : \mathtt{X}\ (\psi)$$

Definition (Sémantique)

- $\langle d, \sigma \rangle \models \mathtt{R}(t)$ if $\sigma(t) \in d(\mathtt{R}), \mathtt{R} \in \mathbf{R}$
- $\langle d, \sigma \rangle \models t_1.\mathtt{A} \square t_2.\mathtt{B}$ if $\sigma(t_1)(\mathtt{A}) \square \sigma(t_2)(\mathtt{B})$
- $\langle d, \sigma \rangle \models t.\mathtt{A} \square \mathtt{c}$ if $\sigma(t)(\mathtt{A}) \square \mathtt{c}$
- $\langle d, \sigma \rangle \models \neg\psi$ if $\langle d, \sigma \rangle \not\models \psi$
- $\langle d, \sigma \rangle \models \psi_1 \wedge \psi_2$ if $\langle d, \sigma \rangle \models \psi_1$ and $\langle d, \sigma \rangle \models \psi_2$
- $\langle d, \sigma \rangle \models \exists t : \mathtt{X}\ (\psi)$ if there exists a tuple $\mathtt{t}$ over $\mathtt{X}$ such that $\langle d, \sigma_{t \mapsto \mathtt{t}} \rangle \models \psi$

Requête TRC

$$Q = \{t \mid \psi(t)\}$$

Réponse à une requête:

$$Ans(Q, d) = \{\sigma(t) \mid \langle d, \sigma \rangle \models \psi\}$$

Safe TRC: restriction syntaxique garantissant l'indépendance du domaine: $Ans(Q, d)$ dépend de d mais pas de $\mathcal{D}$

Idée: aucune valeur ne doit être issue d'une négation

- interdire par exemple $\{t \mid \neg R(t)\}$

Datalog

Extensional DataBase:
$Emp(Empno, Lastname, Workdept, Mgrno)$

Intensional DataBase: $R(n, w, m) \leftarrow Emp(a, n, w, m)$

Requête: $Q(x, y) \leftarrow R(x, w_1, m), R(y, w_2, m), w_1 \neq w_2$

- Datalog $\approx$ Prolog sans symboles de fonction
- Perspective positionnelle w.r.t. perspective nommée du TRC
- Jointure exprimée via des variables partagées

Sémantiques de Datalog

3 sémantiques équivalentes:

- par les modèles
- par les systèmes de preuve (résolution SLD)
- point fixe

Sémantique par les modèles

$$R(n, w, m) \leftarrow Emp(a, n, w, m)$$

$$Q(x, y) \leftarrow R(x, w_1, m), R(y, w_2, m), w_1 \neq w_2$$

Emp			
Empno	Lastname	Workdept	Mgrno
10	SPEN	C01	20
20	THOMP	C01	-
30	KWAN	D21	10
50	GEYER	D21	20
90	HENDER	D21	10

R		
SPEN	C01	20
THOMP	C01	-
KWAN	D21	10
GEYER	D21	20
HENDER	D21	10

Q	
SPEN	GEYER
GEYER	SPEN

Plus petit modèle (au sens de la logique classique)

- Satisfaisant les clauses
- Compatible avec l'EDB

Résolution SLD pour Datalog

$$R(n, w, m) \leftarrow Emp(a, n, w, m)$$

$$\begin{aligned} Q(x, y) \leftarrow & R(x, w_1, m), \\ & R(y, w_2, m), \\ & w_1 \neq w_2 \end{aligned}$$

Emp			
Empno	Lastname	Workdept	Mgrno
10	SPEN	C01	20
20	THOMP	C01	-
30	KWAN	D21	10
50	GEYER	D21	20
90	HENDER	D21	10

$$\dfrac{\dfrac{\text{Emp(10,SPEN,C01,20)}}{\text{R(SPEN,C01,20)}} \quad \dfrac{\text{Emp(90,GEYER,D21,20)}}{\text{R(GEYER,D21,20)}} \quad \text{C01}\neq\text{D21}}{\text{Q(SPEN,GEYER)}}$$

Résultat de la requête: tous les faits Q(,) déductibles

Sémantique de point fixe

$R(n, w, m) \leftarrow Emp(a, n, w, m)$
$Q(x, y) \leftarrow R(x, w_1, m), R(y, w_2, m), w_1 \neq w_2$

Opérateur de conséquence immédiate:

- $P = sch(EDB) + IDB +$ requête
- K : instance de P
- $T_P(K) = \{t \mid \exists\sigma\exists H \leftarrow B \in P : t = H\sigma, B\sigma \in K\} \cup K$

Résultat de la requête: tuples $Q(\ ,\ )$ dans $T_P^\infty(EDB)$.

Récursion et négation

Récursion:

- fonctionne bien avec les 3 sémantiques présentées

Négation:

- Restriction syntaxique: stratification:
  - Attribut de niveau à chaque symbole de relation
  - Négation seulement sur les symboles de niveau strictement supérieur
- Alternative: logique tri-valuée

SPARQL

Langage d'interrogation des graphes RDF.

- Graphe RDF: étiqueté sur les arêtes et les sommets
- Arête = triplet (sujet, prédicat, objet)
- Proche de Datalog dans l'esprit:
  - Jointures exprimées via des variables partagées
  - Une seule relation "triplet(, ,)"
- Requête conjonctive = *basic graph pattern*

```
SELECT ?x,?y WHERE {
  ?e1 rdfs:label ?x.     ?e2 rdfs:label ?y.
  ?e1 :mngr ?m.          ?e2 :mgnr ?m.
  ?e1 :dpt ?d1.          ?e2 :dpt ?d2.
  FILTER(?d1 != ?d2)
}
```

1 Introduction

2 Langages logiques

3 Langages navigationnels

4 Langages algébriques

5 Conclusion

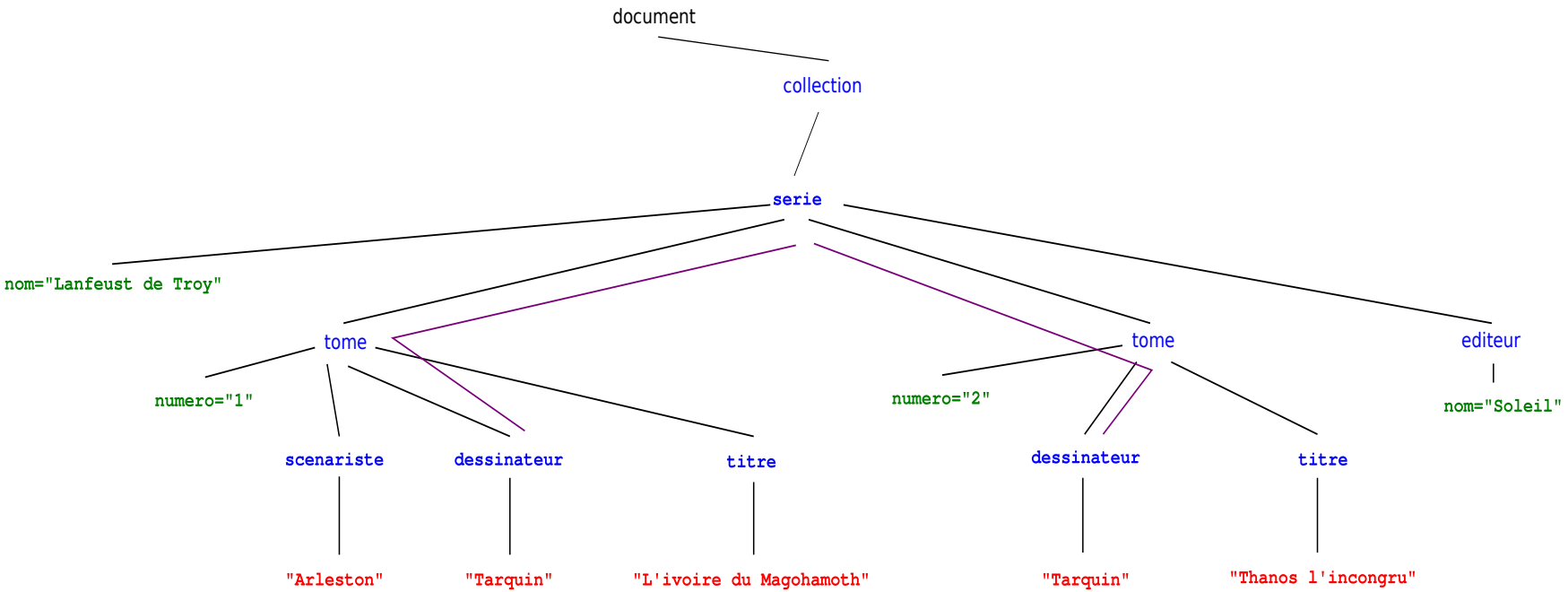
Principe

Dans un arbre/graphe

Spécification de chemin
+
Point de départ:
=
Ensemble de points d'intérêt

Exemple: syntaxe `glob` dans les shells Unix

Exemple dans un arbre XML



"Aller sur un élément `tome`, puis sur un élément `dessinateur`"
Evaluer à partir de l'élément `série`

XPath - expressions de chemin

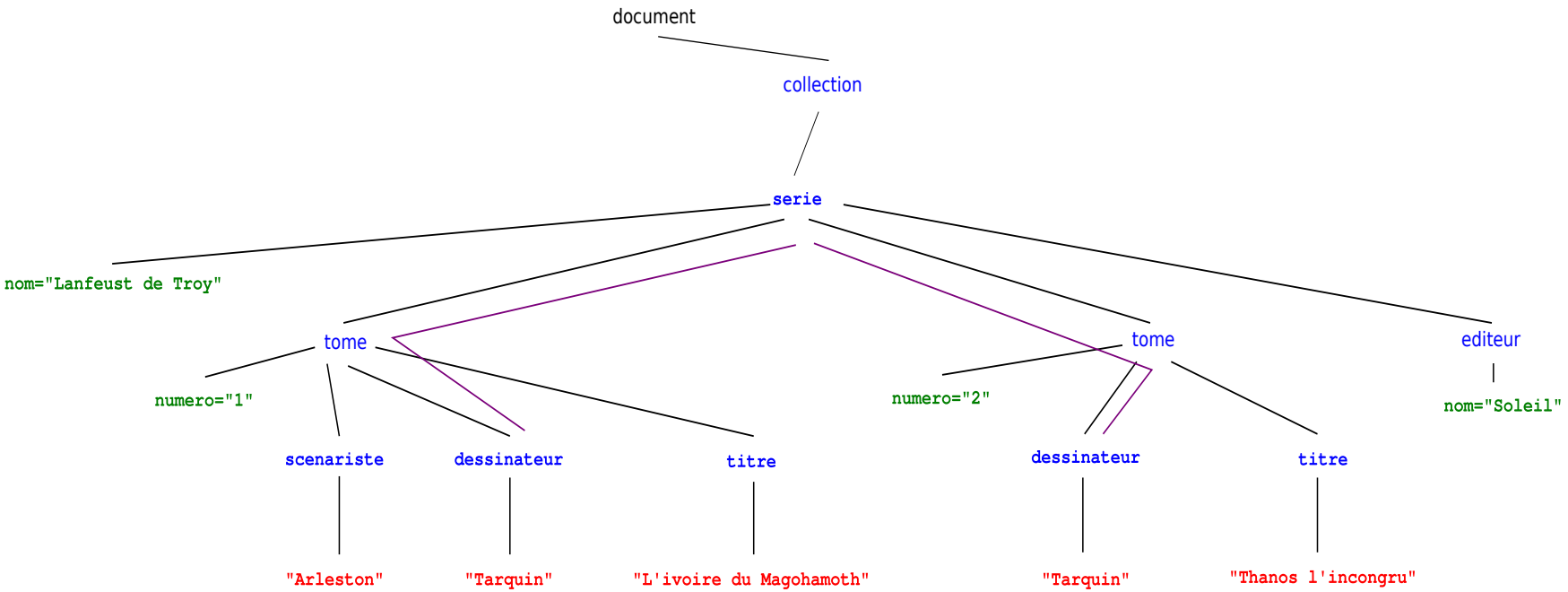
- Suite d'étapes séparée par "/"
- un "/" en début d'expression: départ forcé depuis la racine (document)
- Une étape est de la forme *axe*::*test*[*predicat*]
  - axe: "direction", préselection d'un ensemble de noeud (enfants, frères, ancêtres, etc)
  - test: filtre sur le type (élément, texte, etc) et le nom des noeuds
  - predicat (optionnel): filtrage via une expression booléenne

```
child::element(tome)/child::element(dessinateur)
```

XPath - sémantique

- Pour chaque étape, pour chaque noeud n d'ensemble N_d de noeuds de départs:
  - Calculer N_n^a obtenu en suivant l'axe à partir de n
  - Calculer N_n^t en filtrant N_n^a via le test
  - Calculer N_n^p en filtrant N_n^t via le predicat
- Résultat de l'évaluation de l'étape: $\bigcup_{n \in N_d} N_n^p$

Exemple dans un arbre XML




```
child::element(tome)/child::element(dessinateur)
```

XQuery

Constat: XPath (et les langages navigationnels en général)

- permettent de sélectionner des noeuds
- ne permettent pas de construire de nouvelles structures

XQuery:

- Langage fonctionnel
- Etend XPath :
  - + fonctions
  - + primitives de construction d'arbres XML
  - + FLWOR: pendant du SELECT FROM WHERE de SQL

Property paths en SPARQL

- Chemin spécifié via une *expression rationnelle* de noms d'arête
- possibilité de renverser une arête
- expressivité différente de XPath:
  - \+ expression régulière
  - – tests fins sur les valeurs

Si on suit l'analogie SPARQL $\leftrightarrow$ Datalog:

- Property path $\leftrightarrow$ ensemble de règles récursives

`(:group/^:group)+`

$$:\text{alice} \xrightarrow{:\text{group}} :\text{g1} \xleftarrow{:\text{group}} :\text{bob} \xrightarrow{:\text{group}} :\text{g2} \xleftarrow{:\text{group}} :\text{charlie}$$

Chaînes de personnes se connaissant via la fréquentation d'un même groupe

1 Introduction

2 Langages logiques

3 Langages navigationnels

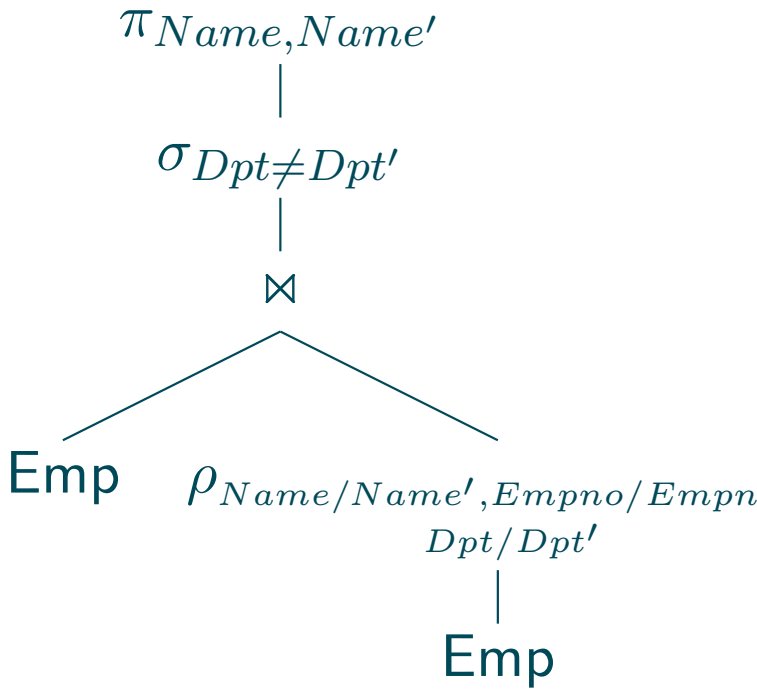
4 Langages algébriques

5 Conclusion

Algèbre relationnelle

Ensemble d'opérateurs sur les relations:

- Sélection de tuples $\sigma_C(R)$
- Projection d'attributs $\pi_{A_1,\ldots,A_n}(R)$
- Jointure $R \bowtie S$
- Opérations ensemblistes: $\times, \cup, \cap, \setminus$
- Renommage: $\rho_{A_1/B_1,\ldots}(R)$



Expressivité: équivalente à celle du TRC autorisé

Optimisation de plans d'exécution

Equivalence entre expressions algébriques:

- $\sigma_C(R \times S) \equiv R \times \sigma_C(S)$
  si C ne porte que sur les attributs de S
- $R \bowtie S \equiv S \bowtie R$ (persp. nommée) et
  $R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$
- ...

Expressions équivalentes, mais coûts d'évaluation différents

- Trouver l'expression optimale (par rapport à un certain modèle de coût) est NP-dur
- Temps alloué pour le trouver fixe $\rightarrow$ solutions approchées

1 Introduction

2 Langages logiques

3 Langages navigationnels

4 Langages algébriques

5 Conclusion

Conclusion

Petit tour d'horizon des paradigmes pour les langages de requête

- importance des langages logiques (+ navigationnels)
  - pour l'utilisateur
- langages algébriques pour l'exécution

Non abordé: agrégation, tri, etc

Quelques problématiques autour des requêtes

- Requêtes en environnements parallèlle / distribués
  - Quels langages pour exprimer des requêtes e.g. sur des architectures type Map/Reduce (e.g. Pig Latin)
  - Modèles/algèbres de calcul et modèles de coût associés
  - Problématiques de placement de données
- Certification des bases de données
  - mécanisation des preuves d'équivalence entre langages (Benzaken et al., ESOP 2014)
- Réécriture de requêtes
  - pour l'intégration et la sécurité (ANR DataCert)
  - pour le nettoyage des données (Mastodon MedClean)
  - pour l'optimisation des requêtes

Références

S. Abiteboul, R. Hull, V. Vianu: *Foundations of databases*
`http://webdam.inria.fr/Alice/`

S. Abiteboul, I. Manolescu, P. Rigaux, M.-C. Rousset, P. Senellart: *Web Data Management*
`http://webdam.inria.fr/Jorge/`

Recommandations W3C: XPATH 2.0, XQUERY 1.0, SPARQL 1.1