Exercice1: Q1 ⊑ Q2

1. Vrai
2. Vrai
3. Vrai
4. Vrai
5. Vrai
6. Vrai
7. Faux
8. Vrai
9. Vrai

Exercice2:

1. L’intégration virtuelle est plus adaptée que l’intégration matérialisée car la fréquence des mises à jour des données est différente, et les ajouts et suppression des relations sont volatiles.

GAV est plus adaptée car Collègue de G n’est pas present dans S1, Connait de G n’est pas present dans S2.

1. Les éléments constituant le système d’intégration

Contexte:

- Sources: relationnel, fichier …

- Interfaces: Sql, NoSql …

Charactéristiques des données: distribué, autonome, hétérogène

Architecture d’intégration: matérialisé, virtuel GAV LAV

Schéma d’intégration:

-Intégration de schéma: définir un schéma global à partir des schémas sources ou

définir des schémas sources à partir du schéma global

-Modèle de données global: relationnel, XML, fichier, …

Mapping:

-Définir la correspondance entre le schema global et les schémas des sources

-Traduire les requêtes, les structures de données

-Analyse les contraintes, les similarités

-Evaluer la puissance, la precision, la maintenance

-Besoin d’ajout de sources

3. Plan d’évaluation des requêtes

a. SELECT C.p1 FROM Connait C JOIN WITH Collègue G ON C.p1=G.p1

1. Identifier les sources de chaque relation dans la requête

Bucket 1 : S1.Personne(p1)

Bucket 2 : S2.MêmeEmployeur(p1,p2)

Bucket 3 : S3. ConnaissancePot(p1,p2)

2. Générer les requêtes candidates en combinant les relations des buckets

Q1(p1) :- S1.Personne(**p1**)

Q2(p1) :- S2.MêmeEmployeur (**p1**,p2)

Q3(p1) :- S3. ConnaissancePot(**p1**,p2)

3. Validation des requêtes candidates :

Q1 ⊑ Q : valide

Q2 ⊑ Q : valide

Q3 ⊑ Q : valide

b. SELECT C.p1 FROM Connait C JOIN WITH Collègue G ON C.p1=G.p1 AND C.p2=G.p2

1. Identifier les sources de chaque relation dans la requête

Bucket 1 : S1.Personne(p1)

Bucket 2 : S2.MêmeEmployeur(p1,p2)

Bucket 3 : S3. ConnaissancePot(p1,p2)

2. Générer les requêtes candidates en combinant les relations des buckets

Q1(p1, p2) :- S1.Personne(**p1**), S1.Personne(**p2**)

Q2(p1,p2) :- S2.MêmeEmployeur (**p1**,**p2**)

Q3(p1,p2) :- S3. ConnaissancePot(**p1**,**p2**)

3. Validation des requêtes candidates :

Q1 ⊑ Q : non valide

Q2 ⊑ Q : valide

Q3 ⊑ Q : non valide

4. Considérer la relation Personne2 (p1, p2) dans la source 1

1. Identifier les sources de chaque relation dans la requête

Bucket 1 : S1.Personne2(p1,p2)

Bucket 2 : S2.MêmeEmployeur(p1,p2)

Bucket 3 : S3. ConnaissancePot(p1,p2)

2. Générer les requêtes candidates en combinant les relations des buckets

Q1(p1,p2) :- S1.Personne(**p1, p2**)

Q2(p1,p2) :- S2.MêmeEmployeur (**p1**,**p2**)

Q3(p1,p2) :- S3. ConnaissancePot(**p1**,**p2**)

3. Validation des requêtes candidates :

Q1 ⊑ Q : valide

Q2 ⊑ Q : valide

Q3 ⊑ Q : non valide

Réécriture Q1 ⋀ Q2 = Q

5. Donner le plan d’évaluation des requêtes par réécriture avec l’algorithme « Minicon »

a. SELECT C1.p1 FROM Connait C1, Connait C2, Collègue G WHERE C1.p1=C2.p2 AND C1.p2=C2.p1 AND G.p1=C1.p1 AND G.p2=C1.p2

Etape 1 : création de MCD

Q2(p1,p2) :- S2.MêmeEmployeur (**p1**,**p2**)

Etape 2 : génération des réécritures

Impossible