Partie SPARQL

Question 1

- On cherche toutes les UE où les deux élèves d'un binôme sont inscrits. (Réponse H)

Question 2

- On cherche tous les éléments ayant pour prédicat `ucbl:departement` et pour objet `ucbl:informatique`. (Réponses D, E, F, H)

Question 3

- On cherche tous les étudiants ayant un binôme et où lui et son binôme ne sont pas inscrits aux mêmes matières. (Réponses C et J)

  etu:2345678 et etu:3456789 sont bien binômes mais sont inscrits tous les deux à ue:mif17

Question 4

- On cherche tous les étudiants inscrits dans au moins une UE. (Réponses A, E, H, I)

  Ici, ce n'est pas précisé que ?a doit être différent de ?b

Question 5

- On cherche tous les étudiants inscrits à une UE où il y a au moins un autre étudiant d'inscrit, sachant qu'ils ne doivent pas être en binôme. (Réponses E et I)

  Contrairement à la question 4, ici il a bien été précisé que ?a et ?b doivent être différents.

Partie MongoDB

Warning : Je fais souvent un raccourci de langage, dans le sens où je dit "l'appel à la fonction reduce" ou "le reduce" mais il faut bien comprendre qu'à chaque appel une fonction reduce tourne pour chaque clé. Ainsi, quand je parle du cas d'un seul reduce, par exemple, cela sous-entend un seul appel à la fonction reduce par clé.

Question 6

- A, B et D (**Fausses**) : Le type du retour n'est pas le même que celui de la partie `valeur` de l'emit. S'il y a un re-reduce, ça va bloquer.

La fonction map renvoie une paire de type : (id, {t: string, c: int})
Le résultat de la fonction reduce doit donc être de type {t: string, c: int}

A renvoie seulement un `string`*, B et D renvoient seulement un* `int`
C renvoie bien le bon type mais selon s'il y a re-reduce ou pas, le résultat ne sera pas le même.

Exemple :

map renvoie les paires (5, {t: "quizz", c: 1}), (2, {t: "exam", c: 1}), (2, {t: "quizz", c: 1}), (5, {t: "homework", c: 1}), (2, {t: "homework", c: 1}), (5, {t: "quizz", c: 1}), (5, {t: "homework", c: 1})

Cas sans re-reduce :

Le reduce assemble les valeurs par clé, ce qui donne

- 5 : [{t: "quizz", c: 1}, {t: "homework", c: 1}, {t: "quizz", c: 1}, {t: "homework", c: 1}]
- 2 : [{t: "exam", c: 1}, 2, {t: "quizz", c: 1}, 2, {t: "homework", c: 1}]

L'application de la fonction reduce renvera donc `s` avec comme valeur:

- Pour 5 : {t: "quizz", c: 16}
- Pour 2 : {t: "exam", c: 9}

Cas avec re-reduce :

Admettons que les paires (5, {t: "quizz", c: 1}), (2, {t: "exam", c: 1}), (2, {t: "quizz", c: 1}), (5, {t: "homework", c: 1}) aillent dans le reduce1 et les paires (2, {t: "homework", c: 1}), (5, {t: "quizz", c: 1}), (5, {t: "exam", c: 1}) dans le reduce2

Le reduce1 assemble les valeurs par clé, ce qui donne

- 5 : [{t: "quizz", c: 1}, {t: "homework", c: 1}]
- 2 : [{t: "exam", c: 1}, 2, {t: "quizz", c: 1}]

L'application de la fonction reduce1 renvera donc `s` avec comme valeur:

- Pour 5 : {t: "quizz", c: 6}
- Pour 2 : {t: "exam", c: 5}

Le reduce2 assemble les valeurs par clé, ce qui donne

- 5 : [{t: "quizz", c: 1}, {t: "exam", c: 1}]
- 2 : [{t: "homework", c: 1}]

L'application de la fonction reduce2 renvera donc `s` avec comme valeur:

- Pour 5 : {t: "quizz", c: 6}
- Pour 2 : {t: "exam", c: 1}

On met ensuite les résultats des deux appels dans un nouvel appel à reduce :

Assemblage des valeurs par clé :

- 5 : [{t: “quizz”, c: 6}, {t: “quizz”, c: 6}]
- 2 : [{t: “exam”, c: 5}, 2, {t: “homework”, c: 1}]

L’application de la fonction renvera donc `s` avec comme valeur:

- Pour 5 : {t: “quizz”, c: 11}
- Pour 2 : {t: “exam”, c: 9}

On voit donc bien que ce résultat n’est pas le même que quand on fait appel à seulement un reduce, ce qui explique que la réponse C est fausse.

Question 7

- C et D (**Fausses**) : Tous les scores sont emit, que ce soit de type “quizz” ou pas, et sans pouvoir savoir à quel étudiant ça correspond, cela ne répond pas à l’énoncé.
- A (**Correcte**) : Dans le map on calcule directement le nombre de scores de type “quizz” (`s`) et on `emit` donc ce nombre.
- B (**Correcte**) : Ici on fait un `emit` par score de type “quizz”.

  Pour A et B, dans tous les cas le reduce fera la somme.

Question 10

- Se référer à l’exemple de la question 6 pour comprendre pourquoi le re-reduce ne fonctionne pas avec la réponse D.

Question 11

- A (**Fausse**) : On veut la note minimale par étudiant et non par UE (`class_id`).
- C (**Fausse**) : De même, ici ça nous triera les données par étudiant par classe, or on veut seulement par étudiant en clé.
- B et D (**Correctes**) : Comme pour la question 7, la B enverra un `emit` par score (on aura donc toutes les notes d’un document), tandis que la D calculera directement la note minimale de l’étudiant dans le document.