Partie MongoDB

Exercice 1

1- Les prédicats des TD seront rdf:type, rel:serie et rdfs:label

Les formes URL sont aussi acceptées, le prof ne note pas l'exactitude syntaxique. (<http://www.w3.org/1999/02/22-rdf-syntax-ns/type>, <http://example.com/rdf/rel/serie> et <http://www.w3.org/2000/01/rdf-schema/label>)

2- De la règle SP on peut déduire les triplets :

- p:benford rdfs:label "Gregory Benford"
- b:fear rdfs:label "Fundation's Fear"
- p:greg rdfs:label "Greg Bear"
- b:chaos rdfs:label "Fundation and Chaos"
- s:sndfund rdfs:label "The Second Fundation"
- p:david rdfs:label "David Brin"
- b:triumph rdfs:label "Fundations's Triumph"

3-

Triplets déduits de Domain :

- p:benford rdf:type t:personne
- p:greg rdf:type t:personne
- p:david rdf:type t:personne

Triplets déduits de Range :

- b:fear rdf:type t:livre
- b:chaos rdf:type t:livre
- b:triumph rdf:type t:livre

Triplets déduits de SC :

- Aucun

Triplets déduits de RSerie :

- b:fear rel:serie s:fundation
- b:chaos rel:serie s:fundation
- b:triumph rel:serie s:fundation

Ainsi la taille de TD est de **16**.

4- Le résultat de la requête sur le graphe saturé est :

- ?b -> b:fear
- ?b -> b:chaos

- ?b -> triumph

5- Il faut rajouter le triplet *rel:serie rdfs:range t:serie*

Exercice 2

1- Ici, on calcule le nombre de produits de la catégorie *conserve* de toute la collection

2-

```
var map2 = function() {
    for (var i in this.produits) {
        emit(this.produits[i].categorie, {minimum: this.produits[i].prix,
            maximum: this.produits[i].prix})
    }
}
```

Une autre solution est d'`emit` la catégorie en clé, avec comme value, un objet contenant min et max avec déjà un tri en amont à l'intérieur du document, mais c'est beaucoup de travail pour pas grand chose :

```
var map2 = function() {
    var tab = new Object()
    for (var i in this.produits) {
        var cat = this.produits[i].categorie
        if (!(cat in tab)) {
            tab[cat] = {min: this.produits[i].prix, max: this.produits[i].prix}
        } else {
            tab[cat].min = tab[cat].min > this.produits[i].prix ?
                this.produits[i].prix : tab[cat].min
            tab[cat].max = tab[cat].max < this.produits[i].prix ?
                this.produits[i].prix : tab[cat].max
        }
    }
    for (var cat in tab) {
        emit(cat, {minimum: cat.min, maximum: cat.max})
    }
}
```

Puis reduce et finalize

```
var reduce2 = function (key, values) {
    var min = values[0].minimum
    var max = values[0].maximum
    for (var i = 1; i < values.length; i++ ) {
        min = min > values[i].minimum ? values[i].minimum : min
        max = max < values[i].maximum ? values[i].maximum : max
    }
```

```
    return {minimum: min, maximum: max}
}

var finalize2 = function(key, value) {
    return value.maximum - value.minimum;
}
```

La première solution que j'avais proposé au tutorat n'est pas recevable. En effet on ne peut pas faire une fonction `map` qui `emit` seulement une paire (catégorie, prix), car la fonction `reduce` traite et renvoie des objets (`{minimum: val1, maximum: val2}`) et pas seulement un nombre. Si la fonction `reduce` traitait un tableau de nombres en entrée, on ne pourrait pas renvoyer un objet, car il serai alors donc impossible de faire un re-reduce. La seconde n'était pas correcte non plus car j'ai fait un `map` par allée et non par catégorie.

3-

```
var map3 = function() {
    for (var i in this.produits) {
        c = []
        a = []
        c[this.produits[i].categorie] = true
        a[this.allee] = true
        emit(this.produits[i].code, {cat: c, allees: a})
    }
}

var reduce3 = function(key, values) {
    var produit = values[0]
    for (var i = 1; i < values.length; i++) {
        for (var c in values[i].cat) {
            produit.cat[c] = true
        }
        for (var a in values[i].allees) {
            produit.allees[a] = true
        }
    }
    return produit
}

var finalize3 = function(key, value) {
    var nbcat = 0
    for (var c in value.cat) {
        nbcat++
    }
    if (nbcat > 1) {
```

```
        res = []
        for (var a in value.allees) {
            res.push(a)
        }
        return res
    } else {
        return undefined
    }
}
```

Exercice 3

```
var mapt = function() {
    var capteur = {count: 1, speeds: this.speed}
    var value = {}
    value[this.capteur] = capteur
    var tranche = tranche_5_min(this.ts)
    //On emit avec la tranche de 5 minutes en clé,
    //et un objet contenant la valeur de la vitesse du capteur courant
    emit(tranche, value)
}
```

Ici, une paire sortant de la fonction map pourrait être

(1577881235, {“rd463-18”: {count: 1, speeds: 65}})

Ainsi un tri par clé du reduce pourrait donner

1577881235 : [{“rd463-18”: {count: 1, speeds: 65}}, {“rd762-04”: {count: 1, speeds: 68}}, {“rd982-11”: {count: 1, speeds: 71}}, . . .]

anotherkey : [{value1: {. . . }}, {value2: {. . . }}, . . .]

. . .

```
var reducet = function(key, values) {
    var res = {}
    for (var value in values) {
        for (var c in value) {
            //Ici, c va prendre la valeur du capteur ("rd463-18" par exemple)
            if (res[c] === undefined) {
                res[c] = {count: 0, speeds: 0}
            }
            res[c].count = res[c].count + v[c].count
            res[c].speeds = res[c].speeds + v[c].speeds
        }
    }
    return res
```

```
}

var finalizet = function(key, values) {
    var moy_max = 0
    var capteur = undefined
    for (var value in values) {
        for (var c in value) {
            moyenne = values[c].speeds/values[i].count
            if (moyenne >= moy_max) {
                moy_max = moyenne
                capteur = c
            }
        }
    }
    return capteur
}
```