Developpement d'Applications pour le Cloud

S9 - IBO

ESILV
nicolas.travers@devinci.fr

Chapitre

Table des matières

Chapitre 1

Avril 2016

Pour l'ensemble de cet examen, nous utiliserons le schéma relationnel suivant et leurs statistiques :

Table & Schéma	Nb tuples
Colis(ID_colis, id_client, Intitule, fournisseur)	4 960 000
Etat_colis(ID_colis, ID_relais, etat, datetime)	29 430 000
Relais(ID_relais, Nom, capacite, localisation)	19 600
Client(ID_client, nom, prenom, adresse)	1 540 000

1.1 Dénormalisation (3 points)

Notre base documentaire contient une collection de documents JSon, reliant le client, le colis et l'ensemble des états associés à ce dernier.

Les états des colis sont : enregistré, dépôt, transit et retrait.

1.1.1 Donner un exemple de document JSon rassemblant l'ensemble de ces données. Nous mettrons l'accent sur la possibilité de distribution des données.

Correction :

```
{  "idColis" : 1,
    "intitule" : "XXX",
    "client" : { "id" : 1001, "nom" : "X", "prenom" : "Y", "adresse" : "XYZ"},
    "fournisseur" : "YYY",
    "etats" : [
        {"etat" : "dépôt", "datetime" : "2016-01-02 10:25:00", "idrelais" : 18},
        {"etat" : "enregistré", "datetime" : "2016-01-02 15:12:00", "idrelais" : 25},
        {"etat" : "transit", "datetime" : "2016-01-03 18:12:00", "idrelais" : 4025},
        {"etat" : "retrait", "datetime" : "2016-01-05 19:03:00", "idrelais" : 4025}
    ]
}
```

1.2 MongoDB et MapReduce (13 points)

Pour les requêtes ci-dessous, exprimez-les sur le document JSon produit dans la question 1.1.1.

1.2.1 (MongoDB API) Donner les noms et prénoms des clients ayant commandé une "tablette" (intitule) (1,5 point)

Correction :

```
db.courses.find({"intitule" : "tablette"}, {"client.nom" : 1, "client.prenom" : 1});
```

Possible avec Aggregate $query + $project

1.2.2 (MongoDB API) Donner les intitulés des colis commandés (état=enregistré) depuis le 1° janvier 2016. (1,5 point)

Correction :

```
db.courses.find({"etats":{"etat":"enregistré", "datetime":{$gte:"01-01-2016"}}, {"intitule" : 1}
```

Attention à bien imbriqué le prédicat de l'état pour que cela soit appliqué au même état (sinon, commande en 2015, livraison en 2016)

1.2.3 (MongoDB API) Donner le nombre de clients ayant commandé un produit "Apple" (fournisseur) (2 points)

Correction :

```
db.courses.aggregate([
        {"$match" : {"fournisseur" : "Apple"}},
        {"$group" : {"_id" : null, "nb" : {"$sum" : 1}}}]);
```

1.2.4 (MongoDB API) Donner par fournisseur, le nombre de colis correspondant dont l'état est de type "retrait". Trier le résultat par le nombre de colis décroissants. (2 points)
Correction :

```
db.courses.aggregate([
        {"$match" : {"etats.etat" : "retrait"}},
        {"$group" : {"_id" : "$fournisseur", "nb" : {"$sum" : 1}}},
        {"$sort" : {"nb" : -1}}]);
```

1.2.5 (MongoDB API & MapReduce) Donner pour chaque point relais (idrelais) et état, le nombre de colis correspondant (3 points)
Correction :

```
var map = function (){
        for(var i=0;i<this.etats.length;i++){
                emit({"idrelais" : this.etats[i].idrelais, "etat" : this.etats[i].etat, 1);
        }
}

var reduce = function (key, values){
        return Array.sum(values);
}
```

1.2.6 (MapReduce) Donner pour chaque client (id_client) : le nombre de colis par type d'état (3 points)
Correction :

```
var map = function (){
        for(var i=0;i<this.etats.length;i++){
                if(this.etats.etat == "enregistre")
                        emit(this.fournisseur, {enregisre:1});
                else if(this.etats.etat == "depot")
                        emit(this.fournisseur, {depot:1});
                else if(this.etats.etat == "transit")
                        emit(this.fournisseur, {transit:1});
                else if(this.etats.etat == "retrait")
                        emit(this.fournisseur, {retrait:1});
        }
}

var reduce = function (key, values){
        var etats = {enregistre:0, depot:0, transit:0, retrait:0};
        for(var i=0;i<values.length;i++){
                if(values[i].enregistre)
                        etats.enregistre += values[i].enregistre;
                if(values[i].depot)
                        etats.depot += values[i].depot;
                if(values[i].enregistre)
                        etats.transit += values[i].transit;
                if(values[i].enregistre)
                        etats.retrait += values[i].retrait;
        }
        return Array.sum(values);
}
```

1.3 Hachage (4 points)

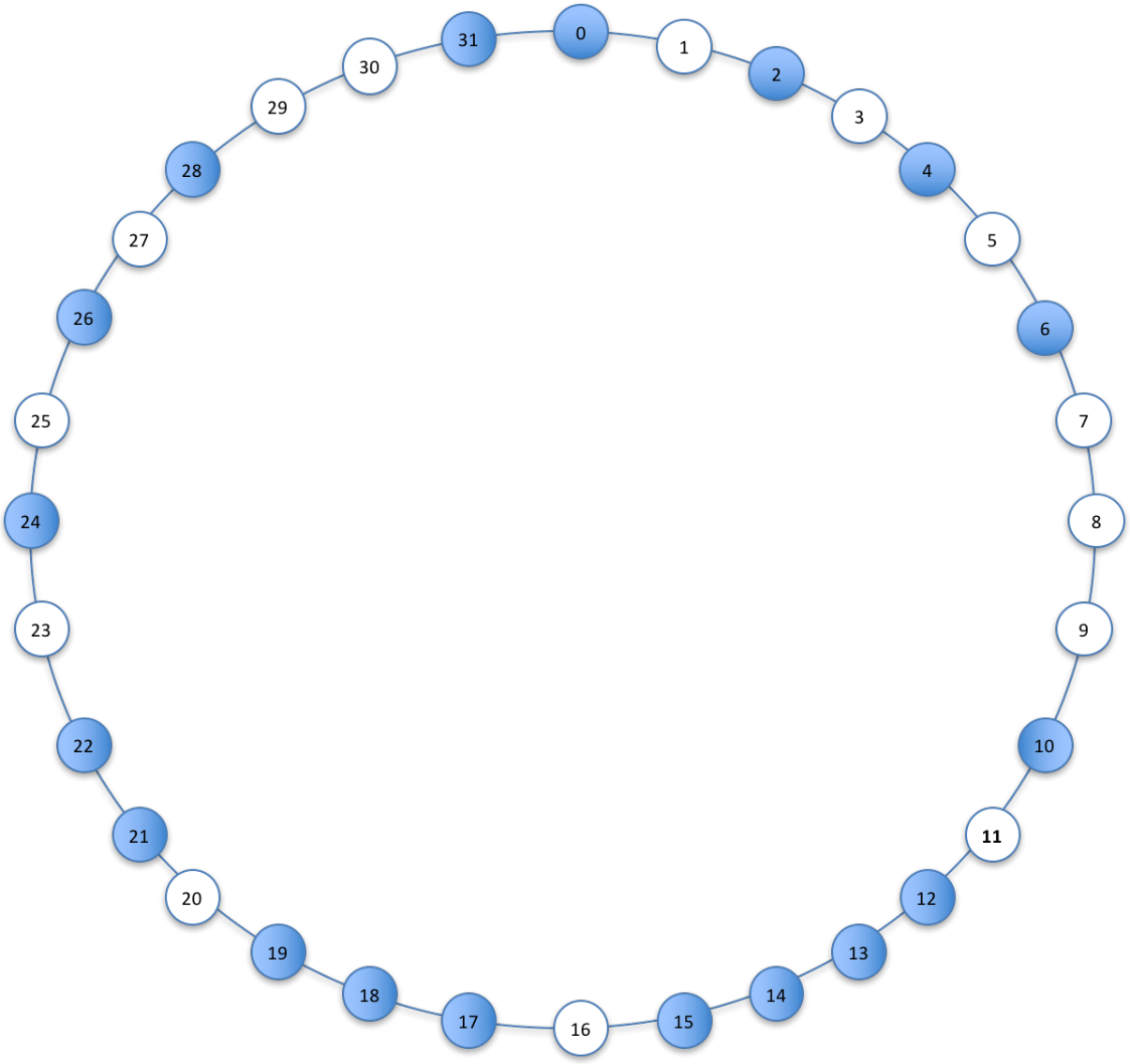
1.3.1 On souhaite créer une table hachage linéaire sur les ID_relais. Nous représenterons les documents uniquement par l'identifiant de metteur en scène. Avec p=0, n=2 et une capacité de 3 par bucket. Faire apparaître les étapes de changement. *(3 points)*
Liste des identifiants : 1, 53, 9, 42, 27, 23, 22, 20, 16, 15, 53, 21, 12, 14, 18, 10, 24, 15, 23, 21, 7
Correction : On commence à p=0 et n=1. On termine à p=3 et n=3.

0	16		
1	1		
2	18		
3	27		
4	20, 12		
5	53, 53, 21	→	21
6	22, 14		
7	23, 15, 15	→	23, 7
8	24		
9	9		
10	42, 10		

2.1 Données

Soit la liste de mots suivante :
`Animaux, Capitales, Couleurs, Départements, Herbes, Eléments, Épices, USA, Fleurs, Légumes, Vêtements, Voitures, Métiers, Arbres, Monnaies, Dinosaures, Transport, Outils, Chaînes, Télévision, Pays, Pokémon, Prénoms , Sports, Fruits, Verbes, Groupe, Irréguliers, Anglais`

Pour placer une donnée, nous utiliserons une fonction de hachage qui prend la première lettre de chaque mot et prendre sa place dans l'alphabet.



2.2 DHT

Soit l'anneau Chord suivant dont les serveurs connectés au réseau sont grisés (0,3,4,6...) :

2.2.1 Placer les mots sur cet anneau (anneau au verso)

2.2.2 Donner la table de hachage des noeuds 4, 10, 19, 26.

2.2.3 Un nouveau serveur se connecte et prend le noeud numéro 8. Déplacer les valeurs associées et barrer les anciennes.

2.2.4 Le serveur 14 tombe en panne, quels sont les serveurs ayant un réplicat (avant la panne) sachant que nous avons un facteur de réplication de 3 ? Mettre une étoile sur les serveurs correspondant.

2.2.5 Modifier les tables de hachage correspondantes à ces deux modifications (barrer l'ancienne et ajouter la nouvelle)

2.2.6 Le serveur 19 reçoit une requête demandant le mot 'Métiers'. Afficher sur l'anneau les étapes de parcours de la requête

Chapitre 3

Mars 2018

Le schéma de base de données suivant sera utilisé pour l'ensemble du sujet. Il permet de gérer une **très grosse** base de données d'un *Store* sur des jeux vidéo pour smartphone.

App	(IdApp, nom, description, icone, image)	500 000t
Connexion	(IdCompte, Datetime, IdApp, duree)	665 000 000 000t
Compte	(IdCompte, Surnom, dateNaissance, avatar)	485 000 000t

3.1 Dénormalisation (6 points)

Nous souhaiterions dénormaliser notre base de données pour optimiser les interrogations suivantes :

— Nom des applications les plus utilisées du moment ;

— Nom des joueurs les plus connectés par application et par mois ;

3.1.1 `(4 points)` Nous souhaiterions dénormaliser notre base de données pour répondre à ces deux requêtes de manière efficace. Proposer un ou deux documents JSon en mettant l'accent sur la distribution et la dynamicité.

3.1.2 `(2 points)` Quel serait la clé utilisée pour distribuer ces données efficacement ? Argumentez votre choix.

Correction :

```
{
  "_id" : {"idCompte":1, "datetime": "2018-03-12 11:01:00", "idApp" : 128},
  "duree" : 3189,
  "app" : {"nom": "SocialChess", "description": "...", "icone": "http://...",  "image": "http://..."},
  "compte" : {"surnom": "Chewbii", "dateNaissance": "1979...", "avatar": "http://"}
}
```

Au vu des requêtes, il serait bon de créer un *partitioning key* sur "_id.idApp" et le *clustering key* sur "_id.idCompte", puis "_id.datetime".

3.2 MongoDB (6 points)

Pour les requêtes ci-dessous, exprimez-les sur la collection "*Connexion*" contenant des documents comme produits dans la question 3.1.1.

3.2.1 (requête de type `aggregate` - 3 points) Nom des 20 applications les plus utilisées (en fonction des durées de connexions) du moment (depuis le dernier mois) ;

Correction :

```
db.connexion.aggregate([
  {$match : {"_id.datetime" : {$gte : "2018-02-14 00:00:00"}}},
  {$group : {"_id" : "$app.nom", "total" : {"$sum" : "$duree"}}},
  {$sort : {"total" : -1}},
  {$limit : 20}
]);
```

3.2.2 (requête de type `MapReduce` - 3 points) Nom des joueurs les plus connectés par application et par mois (tri effectué après le MapReduce) ;

Correction :

```
var map = function () {
  mois = this._id.datetime.substr(5, 2);
  emit ({"joueur" : this._id.idCompte, "app" : this._id.idApp, "mois" : mois}, this.duree);
}
var reduce = function (cle, durees){
  duree = 0;
  for(i=0; i<durees.length; i++){
    duree += durees[i];
  }
  return duree;
}
var queryParam = {"query": {}, "out" : "test"};
db.connexion.mapReduce (map, reduce, queryParam);
db.test.find().sort({"value" : -1})
```

3.3 DHT (8 points)

Soit la liste de mots suivante :
`Animaux, Capitales, Couleurs, Départements, Herbes, Eléments, Épices, USA, Fleurs, Légumes, Vêtements, Voitures, Métiers, Arbres, Monnaies, Dinosaures, Transport, Outils, Chaînes, Télévision, Pays, Pokémon, Prénoms , Sports, Fruits, Verbes, Groupe, Irréguliers, Anglais`

Pour placer une donnée, nous utiliserons une fonction de hachage qui prend la première lettre de chaque mot et prendre sa place dans l'alphabet.

Soit l'anneau Chord suivant dont les serveurs connectés au réseau sont grisés (0,3,4,6...) :

3.3.1 Placer les mots sur cet anneau (anneau au verso)

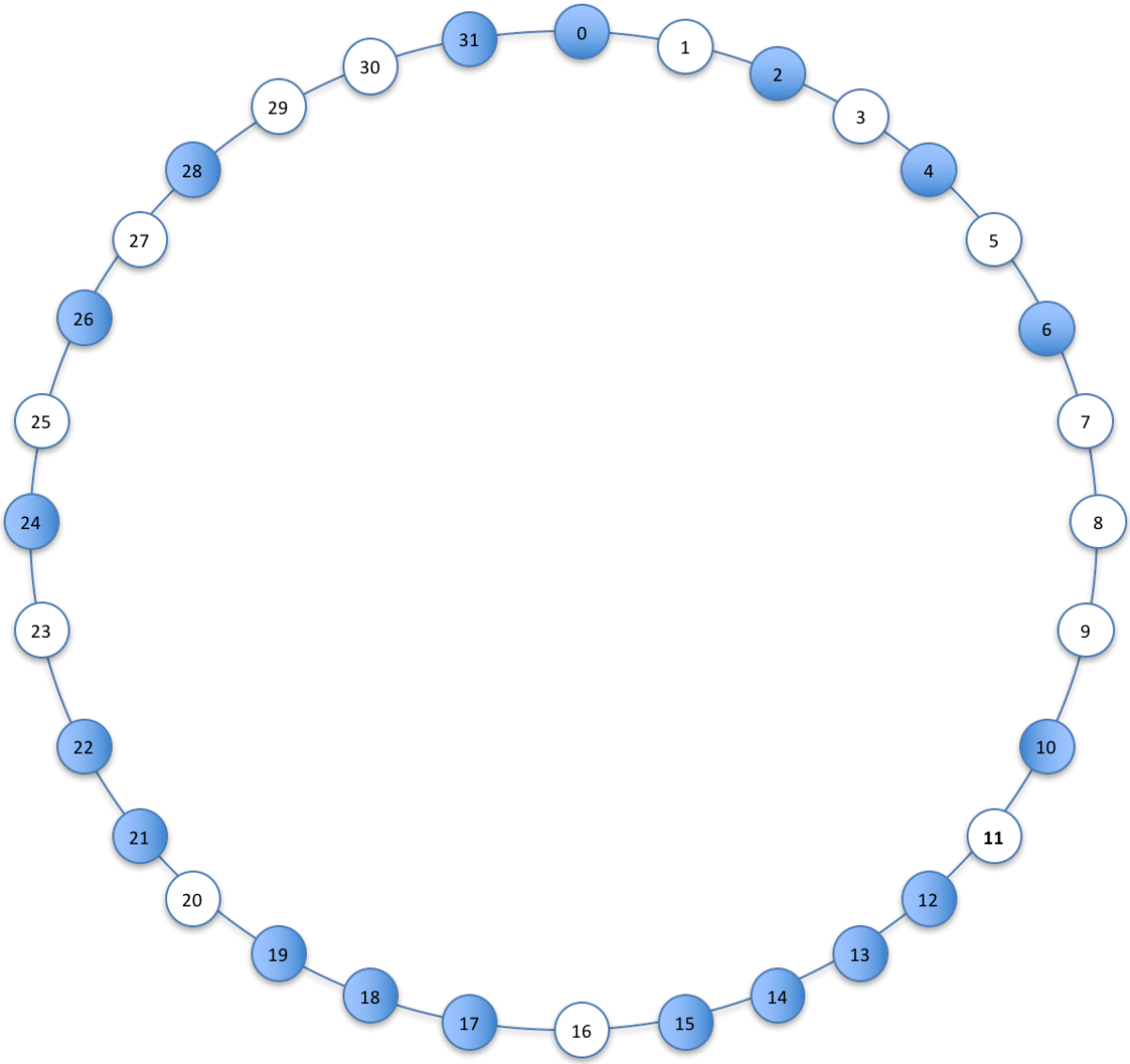
3.3.2 Donner la table de hachage des noeuds 4, 10, 19, 26.

3.3.3 Un nouveau serveur se connecte et prend le noeud numéro 8. Déplacer les valeurs associées et barrer les anciennes.

3.3.4 Le serveur 14 tombe en panne, quels sont les serveurs ayant un réplicat (avant la panne) sachant que nous avons un facteur de réplication de 3 ? Mettre une étoile sur les serveurs correspondant.

3.3.5 Modifier les tables de hachage correspondantes à ces deux modifications (barrer l'ancienne et ajouter la nouvelle)

3.3.6 Le serveur 19 reçoit une requête demandant le mot 'Métiers'. Afficher sur l'anneau les étapes de parcours de la requête


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31