COUCHBASE SERVER Travaux Dirigés

MEHDI LAMRANI – JUILLET 2017

TP 1 : OPERATIONS CRUD

- Enoncé :
  - Modélisation d'un document JSON représentant un USER avec ses attributs (identifiant, type, login, mot de passe)
  - Définition de la clé du document (UUID)
  - Création et connexion au Cluster
  - Connexion au pool via la commande OpenBucket
  - Vérifier la connexion

TP2 : OPERATIONS CRUD

- Enoncé :
  - Ecriture d'un script/programme de stockage conditionnel du document dans Couchbase via les commandes Bucket upsert / insert / replace
  - Gestion de l'Exception keyAlreadyExists.
  - Gestion des Erreurs :
    - NodeJS : Ecriture d'une fonction de Callback pour les exceptions
    - Java : Gestion des Exceptions
  - Accès en lecture d'un document préenregistré avec la méthode get.

TP2 : OPERATIONS CRUD

- Enoncé : Gestion de l’expiration
  - Ecriture d’un programme de gestion du TTL (Expiry)
  - Test avec un Timer, essai de récupération avant et après expiration
  - Gestion de l'Exception keyNotFound.

TP3 : OPERATION CRUD

- Enoncé :
  - Accès en lecture des métadonnées d'un document.
  - Suppression de document par clé
  - Ecriture d'une fonction de Modification et mise à jour de l'expiration
  - Test des fonction touch et getandtouch

TP4 : OPERATIONS BATCH LECTURE

- Enoncé :
  - En utilisant le sample de votre choix
  - Ecrire un programme qui récupère 10 documents par clés spécifiques

    - Node JS
      - Ecriture d'une fonction dédiée en utilisant getMulti

    - Java
      - Ecriture d'une méthode dédiée en utilisant Observable et bucket.async()

    - Génériser le programme avec un nombre de documents paramétrés et une liste de string

TP4 : OPERATIONS BATCH ECRITURE / LECTURE

- Enoncé :
  - En utilisant le sample de votre choix
  - Ecrire un programme qui génère 1000 documents à la volée, avec un Id incrémental

  - Observer ce qui se passe à lecture en Batch de ces 1000 documents (Stats serveur)
  - Comparer à ce qui se passe pour la même opération en mode sérialisé
  - Conslusions ?

TP5 : CONCURRENCE PESSIMISTE AVEC LE CAS

- Enoncé 1 : Deux update successifs
  - Création d'un document User
  - Modification d'un champ du document au choix
  - Ecriture en Base via Replace
  - Récriture du même objet en Base
    - Que constatez-vous ?
    - Pourquoi ?
    - Comment résoudre ce souci ?

  - Effectuer la rectification nécessaire
    - Conclusion ?

TP6 : CONCURRENCE PESSIMISTE AVEC LE CAS

- Enoncé 2.A : Update avec un Lock
  - Récupérer le document précédent avec un Lock
  - Utiliser la valeur maximale de timeout
  - Récupérer le document précédent de façon classique
  - Modification d'un champ du document au choix
  - Effectuer un Replace
    - Que constatez-vous ?
    - Pourquoi ?
    - Comment résoudre ce souci ?
  - Effectuer la rectification nécessaire
    - Conclusion ?

TP7 : CONCURRENCE PESSIMISTE AVEC LE CAS

- Enoncé 2.B : Update avec un Lock
  - Récupérer le document précédent avec un Lock
  - Utiliser la valeur de timeout à N secondes
  - Emettre un Sleep de N+1 secondes
  - Récupérer le document précédent de façon classique
  - Modification d'un champ du document au choix
  - Effectuer un Replace
    - Que constatez-vous ?
    - Pourquoi ?

TP8 : CONCURRENCE PESSIMISTE AVEC LE CAS

- Enoncé 3 : Lock & Multithreading
  - Récupérer le document précédent avec un Lock
  - Ecrire le code de MultiThreading suivant
    (Node JS : Lancer deux programmes concurrents)
  - Créer une boucle qui lance 10 Process
  - Chaque process :
    - Modifie un champ au choix dans le document
    - Efffectue un update

    - Que constatez-vous ?
    - Pourquoi ?
    - Comment résoudre ce souci ?

  - Effectuer la rectification nécessaire
    - Conclusion ?

TP9 : CONCURRENCE AVEC LE MUTATION TOKEN

- Pour aller plus loin :
  INTRODUCTION DES NOUVELLES FONCTIONNALITES
  High Performance Consistency / Enhanced durability
  - Activer les MutationTokens dans l'environnement Client
  - Création d'un document User
  - Modification d'un champ du document au choix
  - Ecriture en Base via Replace 3 fois de suite avec récupération et affichage du Token
  - Affchers une à une les propriétés du Token
    - Que constatez-vous ?
    - Quelle utilité/application cela peut-il avoir ?
    - Quelles sont les régles que l'on peut inférer entre le UUID et le Numéro de Séquence pour discriminer la propagation de la mutation ?
  - Essayer de générer un **DocumentConcurrentlyModifiedException** (avec le CAS, sans Token activé)

TP10 : CREATION D'UNE VIEW

- Enoncé : Création d'une vue
  - En utilisant le Sample Data de votre choix
    - Ecrire un index qui définit une vue « par nom » des documents
    - Vérifier et afficher le résultat via la Web UI

TP11 : CONSULTATION D'UNE VIEW

- Enoncé :
  - Ecrire le code Java / NodeJS suivant :
    1. Appeler la vue créée
    2. Afficher les clés des N premiers Eléments de la vue
    3. Afficher les champs des documents
    4. Afficher les documents commençant par la lettre alphabétique de votre choix
    5. Afficher le nombre documents commençant par la lettre alphabétique de votre choix
    6. Ecrire une boucle qui affiche le nombre de documents pour chaque lettre de l'alphabet

TP12 : AFFICHAGE GRAPHIQUE D'UNE VIEW

- Enoncé :
  - Ecrire le code Java / NodeJS suivant :
    1. Ecrire le résultat du dernier affichage au format TSV, avec comme entête : ''letter frequency'' (sans les guillemets, séparés par un tab)
    2. Ouvrir programmatiquement le fichier D3JS html fourni prenant comme paramètre interne le fichier TSV de sortie
    3. Observer la distribtion des noms de documents choisis dans le diagramme.