# Projet NoSQL

# Partie 1 : booster les performances de la BDD SQL grâce à Redis

1. Implémenter un système de recommandations <- n’était plus à faire
2. Implémenter un système de comptage de vues sur les produits vus par les internautes  
     
   Nous avons créé une classe composée de méthodes statiques permettant de compter le nombre de vue des produits et de stocker le tout sur une basse de données redis.  
   
3. Utiliser Redis comme cache pour améliorer les performances de la base de données SQL  
     
   Pour utiliser un cache de la manière la plus efficace possible, il faut mettre en cache le résultat d’une requête lorsque les cas suivants sont vrais :[[1]](#footnote-1)

* Le rendu de la page implique de nombreuses requêtes de base de données et / ou de logique métier
* La page est visitée fréquemment par vos utilisateurs
* Les données sont les mêmes pour chaque utilisateur
* Les données ne changent pas souvent

C’est pour cela que nous avons décidé d’instaurer le cache sur le menu principal des produits puisqu’elle est consultée régulièrement, elle est la même pour tout le monde et les données ne changent que très peu. De plus, si le cache est placé sur un page produit, les vues ne s’incrémentent pas et ne sont pas suffisamment mise à jour.



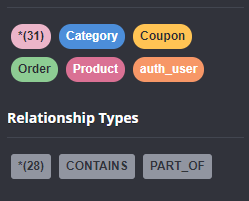


# Partie 2 : Designer une nouvelle base de données NoSQL qui remplacerait la base de données SQLite actuelle

1. Choisir le modèle qui nos convient et justifier les avantages de ce modèle

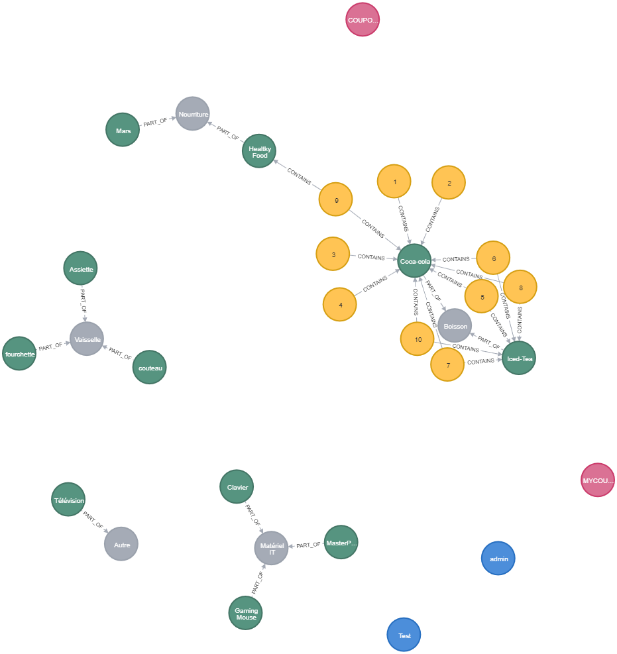
* Nous avons choisi Neo4j car cela permet de faire plus de lien entre les produits et catégories et entre le client et sa commande. Plutôt que MonogDB qui demande la création de nouveaux champs

1. Créer la base de données en y ajoutant les modifications opportunes afin de prendre avantage du modèle choisi.

* Pour créer notre table nous avons créer un fichier qui permet de créer les différentes tables et un autre fichier qui permet de lier les différents types de données ensemble.  
  

1. Implémenter une solution qui permette de transférer automatiquement les données contenues dans la BDD SQLite dans notre nouvelle base de données

* Exportation des données du modèle SQL en fichier JSON  
  
* Une image contenant ustensiles de cuisine, passoire, brosse, alimentation

  Description générée automatiquementCréation des tables  
  
* Création des liens entre les données  
  
* Pour faciliter l’exportation nous avons créé un fichier qui exécute les trois fichiers les uns après les autres afin que tout se fasse automatiquement



# Partie 3 : Intégrer cette nouvelle base de données dans l’application Django du site d’e-commerce

1. Le site doit pouvoir fonctionner comme précédemment
2. Héberger la BDD NoSQL sur un serveur distant

Voici la démarche pour recréer la base de données à distance

1. Se rendre sur le site <https://sandbox.neo4j.com/>
2. Démarrer un nouveau projet vide
3. Dans la section détail de connexion, vous trouverez l’url de la base de données à distance
4. Au niveau de la clé à cote du bouton « Open with bloom » vous trouverez le nom d’utilisateur et le mot de passe
5. Pour avoir une vue de la base de données il suffit de s’y connecter sur « Neo4j Desktop »
6. Pour remplir la base de données avec les bonnes infos il faut modifier deux fichiers « Create\_link » et « Create\_table »
   1. Au niveau de la ligne 4 il suffit de remplacer l’URL par votre url et ensuite modifier les paramètre « auth » par d’abord le nom d’utilisateur et le deuxième élément par le mot de passe.
   2. C’est la même démarche pour les deux fichiers
7. Pour relier le projet à la base de données il faudra modifier les différents fichiers models.
   1. Dans chacun des fichiers models comprenant la ligne :  
        
      Il faut remplacer « neo4j » par votre nom d’utilisateur et « interpreters-flashes-soap » par le mot de passe qui vous a été fourni.  
      Ensuite la partie suivant le « @ » doit être remplacée par les numéros contenu dans votre url.
8. Il vous suffit ensuite de :
   1. Créer votre environnement virtuel et l’activer
   2. Installer tous les packages contenu dans requirements.txt
   3. Créer et lancer votre base de données redis sur ubuntu
   4. Lancer votre projet sur localhost

1. <https://realpython.com/caching-in-django-with-redis/> [↑](#footnote-ref-1)