NoSQL Storage Cost Evaluation

Full Automated Analysis and GUI Application

Authors:
Myriam Ait Said
Mohamed Aymane Sfouli
Luca Rougemont
George Shamieh

ESILV Big Data Structure

Introduction

Dans le cadre du cours *Big Data Structure*, nous avons étudié l'impact des différentes stratégies de dénormalisation NoSQL sur les coûts de stockage et la distribution lors du sharding. Au delà des calculs demandés, nous avons choisi d'aller beaucoup plus loin : construire un outil complet, robuste et totalement générique capable d'analyser n'importe quel schéma NoSQL.

L'objectif initial était académique. Le résultat final est une application professionnelle, portable, élégante et utilisable par n'importe quel utilisateur non technique.

Notre démarche a été guidée par trois idées simples:

1. Le stockage doit être mesuré avec précision, de manière transparente.
2. La manipulation des schémas ne doit pas être réservée aux développeurs.
3. L'outil doit rester générique, élégant, et simple à utiliser.

C'est ainsi qu'est née notre application : **Big Data Structure Storage Estimator**.

1 Architecture du Projet

Notre dossier final n'est pas une simple remise de scripts isolés. Il s'agit d'un **mini projet logiciel structuré**, composé de plusieurs modules clairement définis.

1. AppCore/ — Le cœur de l'application

- **gui_app.py** : l'interface graphique moderne réalisée en Tkinter.
- **stats.json** : les cardinalités globales (produits, stocks, orderlines...).
- **schema/** : tous les schémas utiles au projet (DB1 à DB5, exemple de la prof).
- **bigdata/models.py** : classes et abstractions décrivant les collections.
- **bigdata/sizes.py** : l'algorithme central de calcul de taille.

2. Lancement simplifié

- **Launch_BigData_Tool.vbs** : exécute l'application par un simple double clic, sans console, sans terminal, sans manipulation.

3. Dossier Tests/

Conçu pour valider la généricité de notre solution :

- fichiers JSON de test structurés et non structurés,
- exemple fourni par l'enseignante,
- jeux de données personnalisés permettant d'inférer automatiquement un schéma.

L'application a été testée avec tous ces cas, chacun a parfaitement fonctionné.

2 Fonctionnalités de l'Application

Notre interface graphique offre une expérience fluide, professionnelle et autonome.

2.1 1. Détection automatique des schémas

Toute structure `_schema.json` placée dans `AppCore/schema` apparaît instantanément dans la liste déroulante.

2.2 2. Saisie intelligente du "Document count"

Le champ est rempli automatiquement en fonction du schéma :

- DB1 → nombre de produits,
- DB3 → produits × entrepôts,
- DB4 → nombre d'orderlines.

2.3 3. Analyse complète du stockage

L'outil :

- lit et valide tout JSON Schema,
- calcule la taille exacte d'un document,
- convertit automatiquement en taille de collection et en gigaoctets.

2.4 4. Inférence automatique d'un schéma

Si l'utilisateur charge un `.json` qui n'est pas un JSON Schema, l'application propose automatiquement :

> *"Ce fichier ne correspond pas à un schéma JSON. Voulez vous inférer le schéma automatiquement ?"*

Cette fonctionnalité nous a permis de traiter l'exemple complet fourni par l'enseignante.

3 Les Cinq Schémas Étudiés (DB1 à DB5)

Chaque schéma représente une stratégie différente de dénormalisation :

- **DB1 :** Product avec catégories + supplier.
- **DB2 :** DB1 + information de stock.
- **DB3 :** Stock comme racine, Product complètement embarqué.
- **DB4 :** OrderLine comme racine, avec Product complet.
- **DB5 :** Product embarque toutes les OrderLines liées.

Cette variété nous permet d'observer les avantages et limites de chaque modèle.

4 Règles de Calcul du Stockage

L'algorithme suit strictement les règles du cours :

- Integer / Float : 8 bytes
- String : 80 bytes
- Date : 20 bytes
- LongString : 200 bytes
- Surcoût clé valeur : 12 bytes

Approche algorithmique

Le calcul est récursif :

1. lecture du type,
2. ajout du coût propre,
3. traitement des objets internes,
4. traitement des tableaux,
5. propagation de la taille estimée,
6. multiplication par la cardinalité totale.

5 Résultats du Calcul

Database	Collection	Doc Size (bytes)	Collection Size (GB)
DB1	Product	1096	0.1021
DB2	Product	1372	0.1278
DB3	Stock	1240	23.0968
DB4	OrderLine	1464	5453.8250
DB5	Product	1780	0.1658

6 Interprétation

- **DB1 et DB2** : efficaces, équilibrées.
- **DB3** : duplication massive du Product.
- **DB4** : modèle non viable (plus de 5 To).
- **DB5** : charge maîtrisée malgré la dénormalisation.

7 Sharding

Nos scripts montrent que :

- Un sharding par #IDP est globalement excellent.
- #IDW est très mauvais (fort skew).
- #IDC distribue parfaitement les OrderLines.

8 Validation avec l'Exemple Enseignante

Le JSON de l'enseignante n'était pas un schéma. Grâce à notre inférence automatique, nous avons :

- généré un schéma complet,
- calculé sa taille exacte,
- évalué son comportement au sein de notre pipeline.

Cette étape nous a permis de confirmer la généricité de l'application.

9 Conclusion

Pour cette première partie du projet, nous avons construit une applicatio portable et générale. Elle fonctionne avec n'importe quel schéma, n'importe quel exemple et n'importe quel volume de données.

Ce projet met en lumière :

- notre compréhension des mécanismes de dénormalisation,
- notre maîtrise des problématiques de stockage,
- notre capacité à concevoir des outils fiables et professionnels.