

# Maîtrisez les bases de données NoSQL

🕒 15 hours 📶 Medium

License 

Last updated on 6/4/21



## Choisissez votre base de données NoSQL

### Quels sont les critères de choix d'une BDD NoSQL ?

Maintenant, on se retrouve très embêté car les solutions sont nombreuses, les techniques bien différentes, avec des fonctionnalités variées, des manières différentes de gérer les données où chacun peut trouver son compte. Toutefois, il faut bien faire un choix, et ce choix est crucial !

Le choix de votre système est primordial pour la suite, et il ne faut pas le faire à la légère. En effet, vous allez impacter différentes structures : l'application (connexions, requêtes, fonctionnalités, modélisation des données, gestion...), les développeurs (langage, drivers, méthodologie...), les administrateurs réseaux et systèmes (sécurité, virtualisation, flux, administration), économique (coût d'administration et cloud), et j'en passe. Du coup, qu'est-ce qui va vous faire choisir une technologie plutôt qu'une autre ?

Je vous propose une approche pour prendre une décision, plaise à vous de la suivre. Mais tout d'abord, il nous faut des critères de choix.

#### Le coût

Le **coût** de la solution est un facteur qui peut avoir son importance. Il est bon de consulter l'offre et de voir si une version professionnelle est disponible (payante), les services associés (facilité d'administration/exploitation/déploiement). Le coût d'infrastructure est similaire quelle que soit la solution car dépendant de vos besoins et de l'implémentation des algorithmes que vous aurez réalisés. Toutefois, les offres « Cloud » ne sont pas forcément toutes adaptées à votre choix. Par exemple : DynamoDB n'est disponible que chez Amazon.

#### La cohérence des données

La **cohérence** des données est un critère fondamental pour des applications reposant sur des données « fraîches » ou des applications reposant sur des décisions critiques.

#### La disponibilité du système

À opposer à la **disponibilité** grâce au théorème de CAP, nous pouvons déduire que ces deux critères ne sont pas compatibles. Attention, les temps de latence restent faibles (quelques centaines de millisecondes), mais si cela reste crucial, il ne faut pas hésiter. Toutefois, cela ne concerne que les mises à jour de données déjà existantes, pas les insertions.

#### Le langage d'interrogation

La richesse du **langage** permet également de manipuler simplement la base NoSQL. Il facilite la conduite du changement des développeurs dans le processus d'intégration de la solution. Plus le langage est riche, plus vous serez à même d'exprimer vos besoins. Toutefois, une formation est souvent nécessaire pour comprendre les subtilités du langage.

#### Les fonctionnalités

La **fonctionnalité** recherchée est souvent un critère clé dans le choix. Il est fréquemment à l'origine du changement relationnel/NoSQL. Le but étant de répondre à une demande précise : data mining, recherche textuelle, temps-réel, graphe, outil compatible...

Bien sûr, ces critères ne sont pas exhaustifs mais représentent les principaux choix auxquels nous pouvons faire face. On pourrait imaginer le besoin d'une fonctionnalité supplémentaire qui rajouterait un critère à la liste. On pourrait également rajouter le critère d'administration des machines d'un cluster ; toutefois, grâce à la virtualisation des machines, ce point est automatisable et le critère devient moins important.

<

Immergez vos données dans le NoSQL

>

✓

1. Choisissez votre famille NoSQL

✓

2. Maîtrisez le théorème de CAP

✓

3. Passez à l'échelle


▶

4. Choisissez votre base de données NoSQL

📋


Quiz: Savez-vous vraiment ce qu'est le NoSQL ?

Created by






CentraleSupélec

Grande École d'ingénieurs :  
cycle ingénieur, Master et  
École Doctorale, Mastère  
Spécialisé et formation  
continue



OpenClassrooms, Leading  
E-Learning Platform in  
Europe



## Spécifiez vos besoins



Maintenant que nous avons défini nos critères de choix, il faut prendre une décision. Pour cela, prenez le panel de solutions NoSQL candidates mettez-le dans une **matrice multicritères**. Cette matrice donne en abscisse les critères de choix, et en ordonnée les solutions proposées.

Il faut tout d'abord donner un poids à chacun de ces critères. Nous en avons choisi 5, il suffit maintenant de répartir pour chaque critère une valeur entre 0 et 5. La seule contrainte est que le somme de ces poids soit égale à 5. Un poids de 0 signifie que ce critère n'a aucune importance, et une note supérieure à 2,5 qu'il est primordiale (car il représente à lui seul la moitié des critères).

	Poids	MongoDB	Cassandra	ElasticSearch
Coût	0,5	4	4	4
Cohérence	1	4	2	2
Disponibilité	0,5	2	4	3
Langage	1	4	2	3
Fonctionnalité : Symfony	2	4	2	4
Total	5	19	12	16,5

Il reste maintenant à déterminer pour chaque solution s'il répond ou non à la demande en lui attribuant une **note** (et non un poids). Cette note va de 0 à 4, de manière qualitative (insatisfaisant à satisfaisant) ou quantitative (temps de réponse, volume maximum).

Pour prendre votre décision, il suffit tout naturellement de multiplier vos poids avec les notes. La note finale est sur 20 (5 critères notés maximum sur 4, ce qui fait  $4 \times 5 = 20$ ). La somme la plus grande correspond à la solution donnant le meilleur compromis sur tous vos critères.



Si vous rajoutez des critères, il est conseillé de modifier les notes pour obtenir une borne max de référence (ici 20 est assez naturel).

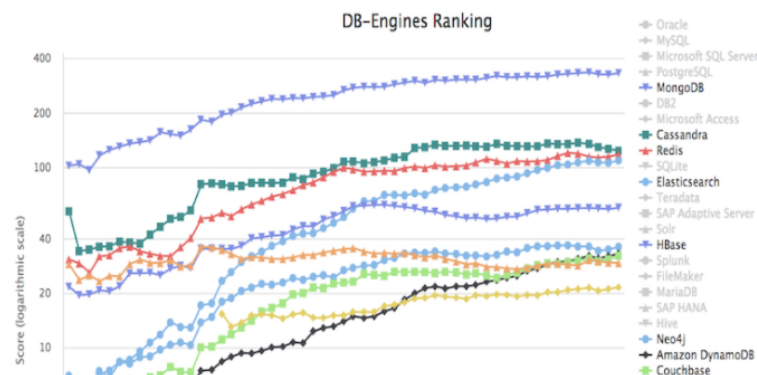
Le tableau donné en exemple parle de lui-même. La pondération d'un projet utilisant le framework *Symfony*, avec gestion de la cohérence et un langage riche oriente le choix vers *MongoDB*. On peut être surpris par le résultat de *Cassandra*, mais le fait qu'il ne soit pas intégré nativement à *Symfony* et son langage trop réduit par rapport à *SQL* en fait une solution non envisageable. De son côté, *Elasticsearch* est une bonne solution, mais est rejetée pour sa gestion de la cohérence et son langage trop complexe.

## En manque d'inspiration ? Allez sur DBEngines



Allez, un petit bonus pour la fin, toujours dans l'idée de permettre de faire votre choix. Le site web [DB-Engines.org](http://DB-Engines.org) donne de nombreuses caractéristiques sur toutes les bases de données référencées sur le Web. Non seulement il donne les caractéristiques pour chaque BDD, mais il évalue également son impact sur la toile.

En répertoriant chaque ticket sur *stackoverflow* (bug, solution, code, conseil...), les blogs et tutoriels, leur utilisation par les entreprises (spécifié dans les rapports d'architecture), *DB-Engines* produit un score pour chaque solution. Le but est de connaître les tendances, aussi bien pour les développeurs que les entreprises. Toutefois, cela n'est qu'un prisme sur le réel, basé uniquement sur les informations présentes sur internet. Mais cela donne déjà un avis et permet de se poser des questions.





Le graphique ci-dessus représente les tendances de janvier 2013 à juin 2017. J'ai volontairement fait un focus sur les solutions NoSQL citées dans ce cours, mais on peut voir aussi (en gris) que les bases de données relationnelles, distribuées et orientées colonnes (sans être NoSQL) sont également prises en compte.

### Commentaires

**MongoDB** reste une solution fortement utilisée par les développeurs. Ce qui en fait sa popularité est son langage puissant et le fait qu'il soit facilement intégrable dans toute application gérant des documents/objets.

**Cassandra** reste en seconde position, même si l'on peut constater une légère baisse ; celle-ci est probablement due au fait que le langage inspiré de SQL est très tentant, mais peut s'avérer assez décevant en raison de ses fortes contraintes. Toutefois, Cassandra reste une très bonne solution pour l'élasticité, le temps de réponse et des fonctionnalités basiques.

**Redis** et **Elasticsearch** sont coude à coude avec Cassandra. Ces deux solutions très tendances permettent de gérer du cache temps réel pour le premier, et des gros volumes de documents textes pour le second. Redis est connu pour sa simplicité, même si la gestion des « facettes » doit être bien pensées. **Elasticsearch** est en particulier utilisé pour faire de la visualisation temps réel et de l'analyse sur des recherches textuelles.

Maintenant que nous avons vu les caractéristiques fondamentales du NoSQL, le moyen de les comparer aussi bien sur les aspects techniques que les fonctionnalités, vous pouvez faire votre choix qui doit être axé sur vos besoins !

I FINISHED THIS CHAPTER. ONTO THE NEXT!

< PASSEZ À L'ÉCHELLE

QUIZ: SAVEZ-VOUS VRAIMENT CE QU'EST, LE NOSQL ? >

### Teachers



**Régis Behmo**

Expert en machine learning, développeur fullstack, grimpeur invétéré et gros, très gros amateur de nouilles chinoises.



**Nicolas Travers**

Maitre de Conférences en informatique au CNAM Spécialités : Bases de données, Optimisation BDD, SQL, NoSQL

#### OPENCLASSROOMS

What we do  
Apprenticeship  
Path experience  
Our blog

#### BUSINESS SOLUTIONS

Business

#### CONTACT



FAQ

#### LEARN MORE

Work at OpenClassrooms  
Become a mentor ☐  
Our store  
Terms of use  
Privacy policy  
Accessibility

English

