DEFINITION RESUMEE

Le NoSQL est un type de bases de données, dont la spécificité est d’être non relationnelles. Ces systèmes permettent le stockage et l’analyse du Big Data.

DEFINITION INTERMEDIAIRE

* Adaptable à de nombreux modèles de données : formats avec des clés, des documents, des colonnes , des graphiques ;
* Alternative aux bases de données placées dans des tables ;
* Utiles pour travailler avec de grands ensembles de données.

NoSQL est une approche de la conception de bases de données qui peut s’adapter à une grande variété de modèles de données, y compris les formats avec des clés, des documents, des colonnes et des graphiques (Graph Database). NoSQL, qui signifie « not only SQL », est une alternative à la base de donnée relationnelle traditionnelles dans lesquelles les données sont placées dans des tables et le schéma de données est soigneusement conçu avant la construction de la base de données. Les bases de données NoSQL sont particulièrement utiles pour travailler avec de grands ensembles de données distribuées.

DEFINITION PLUS DETAILLEE

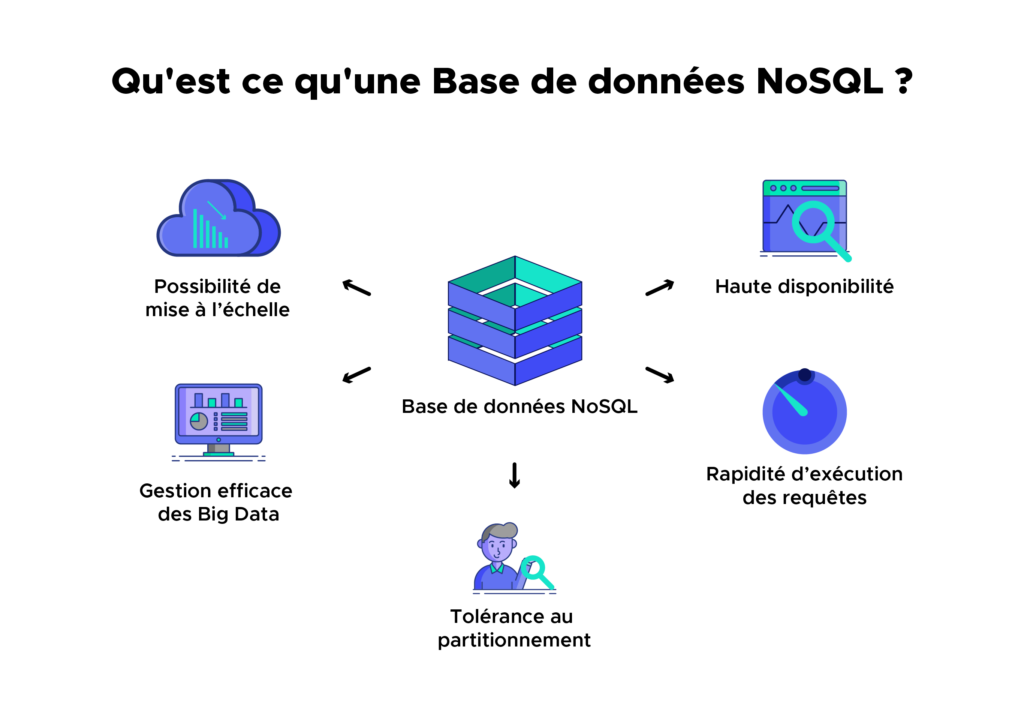
À l’heure du Big Data, les bases de données relationnelles ne sont plus adaptées. Pour prendre en charge les immenses volumes de données, les stocker et les analyser, il est impératif de s’en remettre à de nouvelles solutions.

Une base de données NoSQL est une base de données “non relationnelle”. Il est possible d’y stocker des données sous une forme non structurée, sans suivre de schéma fixe. Les jointures ne sont plus nécessaires, et le scaling est facilité.

On utilise notamment les bases de données NoSQL pour les Data Stores distribués aux besoins élevés en capacité de stockage. Ainsi, NoSQL est utilisé pour le Big Data et les applications web en temps réel. Les géants de la technologie comme Twitter, Facebook ou Google collectent chaque jour plusieurs terabytes de données sur leurs utilisateurs.

Le terme ” NoSQL ” signifie en fait ” Not Only SQL “ (pas seulement SQL). En effet, les bases de données relationnelles utilisent la syntaxe SQL pour le stockage et l’analyse de données.

Ce n’est pas le cas d’une database non-relationnelle. Les systèmes NoSQL sont compatibles avec une large variété de technologies permettant le stockage de données structurées, non structurées, semi-structurées ou polymorphique.



HISTOIRE DE NOSQL

Le terme et le concept NoSQL furent inventés en 1998 par Carl Strozz, afin de désigner sa base de données relationnelle légère et open source. Ce concept a ensuite été adopté et popularisé par les GAFAM tels que Google, Facebook ou Amazon confrontés à d’immenses volumes de données. Les bases de données relationnelles étaient devenues trop lentes.

Plutôt que de mettre à jour leur équipement informatique pour accroître les performances des RDBMS (Relational Database Management System), les géants de la technologie ont choisi de distribuer la charge sur de multiples serveurs hôtes. C’est la méthode dite du ” scaling out “. Les bases de données NoSQL sont idéales pour le scaling-out, puisqu’elles sont non relationnelles.

En l’an 2000, la base de données graphique Neo4j fut lancée. Ce fut ensuite le tour de la Google Bigtable, en 2004, puis CouchDB en 2005. L’histoire des bases de données NoSQL fut aussi marquée par Amazon Dynamo en 2007.

Puis, en 2008, Facebook rend open source la base de données non-relationnelle qu’elle utilise en interne : Cassandra. Cet outil devient la référence des databases NoSQL, et remet le terme NoSQL sous le feu des projecteurs en lui donnant son sens et sa popularité actuelle.

LES CARACTERISTIQUES DE NOSQL

La principale particularité des bases de données NoSQL est qu’elles ne suivent pas le modèle relationnel et ne présentent pas de tableaux sous forme de colonnes fixes. Ces bases de données ne nécessitent pas de normalisation de données ou de mapping relationnel. Il est possible d’interagir sans utiliser de langages de requête complexe.

Une autre particularité est l’absence ou la flexibilité des schémas. Il n’est pas nécessaire de définir de schéma des données, et les données de différentes structures peuvent donc être regroupées sur un même système.

Les bases de données non relationnelles se distinguent aussi par une interface simple d’utilisation pour le stockage et la requête de données. Des APIs permettent de manipuler les données avec diverses méthodes de sélection. Les protocoles, basés sur le texte, reposent principalement sur HTTP REST avec JSON. On utilise en général un langage de requête NoSQL.

La dernière caractéristique d’une base de données NoSQL est d’être distribuée. De multiples bases NoSQL peuvent être exécutées de façon distribuée, offrant des capacités d’auto-scaling et de fail-over. Le concept ACID peut être délaissé au profit de l’élasticité et des performances.

LES DIFFERENTS TYPES DE BASES DE DONNEES NOSQL

On distingue quatre principaux types de bases de données NoSQL : paire clé / valeur, orientée colonne, orientée graph, et orientée document. Chacune de ces catégories a un attribut unique et des limites spécifiques.

Toutefois aucun de ces quatre types de bases de données ne permet de résoudre n’importe quel problème. Il est nécessaire de choisir la base de données adéquate en fonction du cas d’usage.

Dans le cas des bases de données de type paire clé / valeur, les données sont stockées sous forme de paires clé / valeur. Ceci permet la prise en charge de larges volumes de données et de charges lourdes. Les données sont entreposées dans un tableau de ” hash ” au sein duquel chaque clé est unique. La valeur peut être un JSON, un objet BLOB, une ligne de code ou autre.

* Le but de la famille clé-valeur est l'efficacité et la simplicité. Un système clé-valeur agit comme une énorme table de hachage distribuée sur le réseau. Tout repose sur le couple Clé/Valeur. La clé identifie la donnée de manière unique et permet de la gérer. La valeur contient n'importe quel type de données.
* Le fait d'avoir n'importe quoi implique qu'il n'y ait ni schéma, ni structure pour le stockage. D'un point de vue de bases de données, il n'y a pas la possibilité d'exploiter ni de contrôler la structure des données et de fait, pas de langage (SQL = **Structured** Query Language). En soit ce n'est pas un problème si vous savez ce que vous cherchez (la clé) et que vous manipulez directement la valeur.
* Quels types d'applications pourraient correspondre à cette solution ? Voici quelques idées qui pourraient orienter votre choix : détection de fraude en temps réel, IoT, e-commerce, gestion de cache, transactions rapides, fichiers de logs, chat.

Ce type de base de données est le plus basique. Il permet au développeur de stocker plus facilement des données sans schéma. En guise d’exemples, on peut citer Redis ou Dynamo. D’ailleurs, Amazon Dynamo est le modèle initial de cette catégorie de database.

Les bases de données orientées colonnes, comme leur nom l’indique, repose sur des colonnes. Elles sont basées sur le modèle BigTable de Google. Chaque colonne est traitée séparément, et les valeurs sont stockées de façon contigüe.

Cette catégorie de base de données offre de hautes performances pour les requêtes d’agrégation comme SUM, COUNT, AVG et MIN. Pour cause, les données sont déjà disponibles et prêtes dans une colonne. En guise d’exemples, on peut citer HBase, Cassandra ou Hypertable.

* Traditionnellement, les données sont représentées en ligne, représentant l'ensemble des attributs. Le stockage orienté colonne change ce paradigme en se focalisant sur chaque attribut et en les distribuant. Il est alors possible de focaliser les requêtes sur une ou plusieurs colonnes, sans avoir à traiter les informations inutiles (les autres colonnes).
* Cette solution est très adaptée pour effectuer des traitements sur des colonnes comme les agrégats (comptage, moyennes, co-occurences...). D'une manière plus concrète, elle est adaptée à de gros calculs analytiques. Toutefois, cette solution est beaucoup moins appropriée pour la lecture de données spécifiques comme pour les clés/valeurs.
* Quelques exemples d'applications : *Comptage (vote en ligne, compteur, etc), journalisation, recherche de produits dans une catégorie, reporting à large échelle.*
* Les bases de données orientées colonnes, comme leur nom l’indique, repose sur des colonnes. Elles sont basées sur le modèle BigTable de Google. Chaque colonne est traitée séparément, et les valeurs sont stockées de façon contigüe.
* Cette solution est très adaptée pour effectuer des traitements sur des colonnes comme le comptage, les moyennes, etc. D'une manière plus concrète, elle est adaptée à de gros calculs analytiques. Cependant, cette solution est beaucoup moins appropriée pour la lecture de données spécifiques comme pour les clés/valeurs.
* Quelques exemples d'applications : *Comptage (vote en ligne, compteur, etc), journalisation, recherche de produits dans une catégorie, reporting à large échelle.*

Les bases de données Graph-Based stockent les entités et les relations entre ces entités. L’entité est stockée sous forme de noeud, et les relations comme bordures. Il est ainsi facile de visualiser les relations entre les noeuds. Chaque noeud et chaque bord ont un identifiant unique.

Ce type de base de données est multirelationnel. On l’utilise principalement pour les réseaux sociaux, la logistique ou les données spatiales. Parmi les exemples les plus populaires, on peut citer Neo4J, Infinite Graph, OrientDB et FlockDB.

* Les trois premières familles NoSQL n'adressent pas le problème de corrélations entre les éléments. Prenons l'exemple d'un réseau social : dans certains cas, il devient très complexe de calculer la distance entre deux personnes non directement connectées. Et c'est ce type d'approche que résolvent les bases orientées Graphe.
* Dans la base orientée graphe, les données stockées sont : les nœuds, les liens et des propriétés sur ces nœuds et ces liens. Les requêtes que l'on peut exprimer sont basées sur la gestion de chemins, de propagations, d'agrégations, voire de recommandations. Toutefois, contrairement aux solutions précédentes la distribution des nœuds sur le réseau n'est pas triviale.
* Quelques idées d'applications reposant sur des bases orientées graphes : réseaux sociaux (recommandation, plus court chemin, cluster...), réseaux SIG (routes, réseau électrique, fret...), web social (Linked Data).
* Dans la base de données orientée graphe, les données sont stockées sous forme de nœuds, de liens, et de propriétés sur ces nœuds et ces liens. Ce type de base de données est multidirectionnel
* Cependant, contrairement aux autres formes, la distribution des nœuds sur le réseau n’est pas triviale.
* Quelques idées d'applications reposant sur des bases orientées graphes : *réseaux sociaux (recommandation, plus court chemin, cluster...), réseaux SIG (routes, réseau électrique, fret...), web social (Linked Data).*

Les bases de données orientées document stockent et retrouvent elles aussi les données sous forme de paire clé-valeur. Toutefois, la valeur est stockée sous forme de document au format JSON ou XML. La valeur est ainsi comprise par la base de données et peut être trouvée à l’aide d’une requête.

Ce type de base de données offre donc une flexibilité accrue. Il est principalement utilisé pour les systèmes CMS, les plateformes de blogging, ou les applications de e-commerce. En revanche, il ne convient pas pour les transactions complexes nécessitant des opérations ou des requêtes multiples sur des structures agrégées variables. Les exemples les plus connus dans cette catégorie sont Amazon SimpleDB, CouchDB, MongoDB, Riak, Lotus Notes et MongoDB.

* Les bases de données orientées document stockent et retrouvent elles aussi les données sous forme de paire clé-valeur. Toutefois, la valeur est stockée sous forme de document au format JSON ou XML. La valeur est ainsi comprise par la base de données et peut être trouvée à l’aide d’une requête.
* Ce type de base de données offre donc une flexibilité accrue. Il est principalement utilisé pour les systèmes CMS, les plateformes de blogging, ou les applications de e-commerce. En revanche, il ne convient pas pour les transactions complexes nécessitant des opérations ou des requêtes multiples sur des structures agrégées variables.
* Les exemples les plus connus dans cette catégorie sont Amazon SimpleDB, CouchDB, MongoDB, Riak, Lotus Notes et MongoDB.
* Les bases orientées documents sont celles qui sont le plus proche des bases de données classiques, Le but de ce stockage est de manipuler des documents qui contiennent des informations avec une structure complexe en termes de types, de listes, et d’imbrications, Il se base sur le principe de clé / valeur, mais avec plus de champs que cette dernière,
* L’avantage est d’être structurée pour chaque valeur composant le document, Ainsi on peut faire des opérations complexes sur chaque élément du document comme dans une base de données traditionnelle tout en étant dans un système distribué.
* Quelques exemples d'utilisation : *gestion de contenu (bibliothèques numériques, collections de produits, dépôts de logiciels, collections multimédia, etc.), framework stockant des objets, collection d’événements complexes, gestion des historiques d’utilisateurs sur réseaux sociaux.*

AVANTAGES ET INCONVENIENTS NOSQL

NoSQL présente de nombreux avantages, mais aussi des inconvénients. Ces bases de données sont idéales pour le stockage et l’analyse Big Data, et évitent aussi un point de défaillance unique.

Elles facilitent la réplication, et ne nécessitent pas de couche de caching séparée. Les performances sont élevées, et une scalabilité horizontale est possible. Les databases NoSQL peuvent prendre en charge les données structurées ou non structurées de la même manière.

En outre, la programmation orientée objet est facile d’utilisation et flexible. Les bases de données NoSQL ne nécessitent pas non plus de serveur dédié à hautes performances. Elles sont compatibles avec les principaux langages de programmation. L’implémentation est plus simple qu’avec les RDBMS. Le schéma flexible peut être altéré facilement sans interruption.

Néanmoins, ce type de base de données présente aussi des points faibles. On peut citer l’absence de règles de standardisation et les capacités de requêtes limitées. Les capacités de bases de données traditionnelles, comme la consistance lorsque de multiples transactions sont effectuées simultanément, peuvent aussi manquer.

Par ailleurs, il devient difficile de maintenir des valeurs uniques en guise de clés lorsque le volume de données augmente. Ce modèle ne fonctionne pas aussi bien pour les données relationnelles. La courbe d’apprentissage peut être difficile pour les nouveaux développeurs, et les options open source ne sont pas toujours populaires au sein des entreprises. De manière générale, les bases de données relationnelles et leurs outils sont plus matures, plus aboutis et donc plus adoptés.