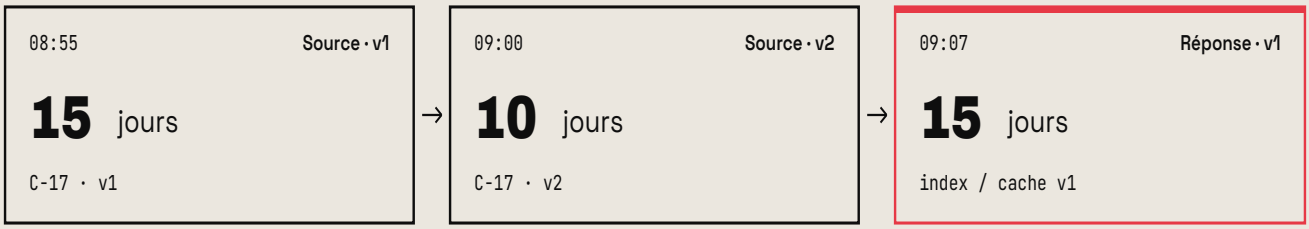
DOSSIER TECHNIQUE · RAG

Base à jour, IA en retard : piloter la fraîcheur du RAG

Une source à jour ne suffit pas à garantir une réponse actuelle. Ce dossier suit le document fictif C-17 de la source jusqu'à la réponse et propose quatre exercices séparés : correction, suppression, révocation et reprise du traitement.

C-17 : suivre la version



Exemple fictif

Préparé le 2 novembre 2026

URL canonique cible

https://datanalysis.re/ressources/artificial-intelligence/fraicheur-rag-source-index-cache

Points à retenir

01 La fraîcheur se vérifie sur toute la chaîne : source, extraction, index, contexte et cache.

02 L'indexation de MongoDB Search et Vector Search est asynchrone ; les caches applicatifs suivent leurs propres règles.

03 Une citation utile identifie la version effectivement utilisée, le passage et le moment de consultation.

04 Correction, suppression et révocation sont des scénarios distincts ; une réponse actuelle peut rester interdite à un utilisateur.

05 La reprise doit conserver les tentatives tout en maîtrisant leurs effets. Deux tentatives pour un effet est ici un attendu fictif, pas une garantie produit.

L'ARTICLE INTÉGRAL

Une réponse peut citer le bon document et utiliser une version dépassée. Dans un système de RAG, où des contenus sont retrouvés puis fournis au modèle, la fraîcheur dépend de toute la chaîne de traitement. La date de modification de la source donne un premier repère. Il faut aussi pouvoir retrouver la version indexée, le contexte transmis et la réponse éventuellement conservée en cache.

Voici une méthode de revue de cette chaîne. Le cas présenté est fictif : il sert à définir les contrôles attendus, sans décrire un incident client ni un test de production.

Commencer par une chronologie vérifiable

À 08 h 55, la version v1 du document C-17 indique un délai de traitement de quinze jours. À 09 h 00, une correction publie la version v2 : le délai applicable est de dix jours. À 09 h 07, une réponse restitue encore quinze jours et cite C-17.

Dans notre scénario, l'index et le cache utilisés reposent encore sur v1. La référence documentaire est donc reconnaissable, mais la réponse ne correspond plus à la version en vigueur.

Avant de retoucher le prompt, il faut établir le chemin réellement suivi. Quelle extraction a alimenté l'index ? Quelle version a été retrouvée ? Le modèle a-t-il été appelé ou une réponse antérieure a-t-elle été réutilisée ? Sans ces traces, plusieurs causes restent indiscernables.

Cartographier les copies qui participent à la réponse

Nous proposons de dresser un inventaire simple : source métier, texte extrait, fragments, embeddings, index de recherche, contexte assemblé, cache de réponse et historique affiché à l'utilisateur. Chaque élément doit avoir une identité, une version connue et un responsable de son actualisation.

Cette cartographie évite de traiter « la base » comme un objet unique. Une source corrigée peut coexister avec un fragment ancien. Un nouvel index peut coexister avec un cache encore valide selon sa propre règle d'expiration. Un historique peut afficher une ancienne réponse à titre d'archive, sans qu'elle doive redevenir une réponse actuelle.

Pour chaque copie, la question utile est concrète : quel événement la remplace, l'invalide ou interdit sa lecture ?

Définir une fraîcheur adaptée à l'usage

MongoDB documente une indexation asynchrone et une cohérence éventuelle pour MongoDB Search et Vector Search. Une écriture dans la collection n'implique donc pas sa disponibilité immédiate dans une requête de recherche. 1 2

Cette propriété doit entrer dans la conception du service. Une base de documentation interne et une instruction opérationnelle urgente peuvent accepter des délais différents. Il faut définir, pour chaque usage, le retard admissible et le comportement attendu lorsqu'il est dépassé.

Nous recommandons de conserver plusieurs horodatages : modification de la source, prise en charge du changement, mise à jour de la représentation recherchable et génération de la réponse. Une mesure unique masque les étapes en retard. Si la fraîcheur nécessaire ne peut pas être établie, l'application peut refuser de conclure, demander une vérification ou signaler explicitement la limite.

Il faut également vérifier le mode de gestion des embeddings. Lorsqu'ils sont produits par l'application, leur renouvellement appartient à son pipeline. Une modification du texte ne garantit pas, à elle seule, leur recalcul.

Donner au cache des conditions de validité

Un cache réduit le travail répété, mais sa réutilisation doit dépendre du contexte. Pour notre document C-17, conserver seulement la question comme clé serait insuffisant : la même question peut appeler une réponse différente après la publication de v2.

Selon l'architecture, la décision de réutiliser une réponse peut prendre en compte la version des documents, le périmètre autorisé, la version des règles métier et une durée maximale. Ce sont des choix de conception à tester, pas des propriétés automatiques du moteur de recherche.

Une durée d'expiration limite la conservation. Elle ne remplace pas nécessairement l'invalidation déclenchée par une correction critique ou un changement de droits. Il faut aussi prévoir le cas où le signal d'invalidation arrive en retard : quelle vérification empêche alors de servir une réponse devenue inacceptable ?

Traiter la suppression comme un parcours complet

Prenons maintenant une variante distincte : à 09 h 10, C-17 est supprimé de la source. Le contrôle ne s'arrête pas à la réussite de cette opération. Il faut examiner les copies utilisées pour retrouver le document, construire le contexte et réutiliser une réponse.

Les change streams MongoDB exposent des événements de suppression avec l'identité du document. Le document complet n'est plus présent dans l'événement standard ; des préimages peuvent être configurées sous certaines conditions. 3

L'application doit donc connaître les dépendances permettant de retrouver ses fragments et ses caches. Nous conseillons de tester la suppression depuis le parcours utilisateur final : recherche, question reformulée, réponse en cache et ouverture d'une ancienne citation. Une copie conservée pour un besoin d'archive doit avoir des règles explicites et ne pas être confondue avec une source encore utilisable pour répondre.

Vérifier les droits indépendamment de la fraîcheur

La révocation constitue une autre variante, à tester séparément. C-17 existe toujours, sa version v2 est correcte, mais l'utilisateur qui pose la question n'a plus le droit de le consulter.

La réponse peut alors être parfaitement à jour et rester interdite pour cette personne. Le contrôle d'accès doit s'appliquer au contenu susceptible d'entrer dans le contexte, ainsi qu'aux réponses déjà mémorisées. Demander au modèle de respecter les permissions ne remplace pas une décision d'autorisation imposée par l'application.

MongoDB décrit notamment l'utilisation d'un champ de tenant comme préfiltre de recherche. 4 C'est un mécanisme utile lorsque l'architecture le prévoit. Il faut encore déterminer le tenant autorisé côté système et traiter les permissions plus fines. Un filtre fourni librement par le demandeur ne constitue pas, à lui seul, une politique d'accès.

Faire porter à la citation une version

Un lien fonctionnel prouve que la ressource peut être ouverte dans un certain contexte. Il ne prouve pas que le passage cité appartient à la version actuellement applicable ni que le destinataire peut le consulter.

Pour une citation exploitable, nous recommandons de conserver l'identifiant documentaire, sa version, le passage utilisé et la date de consultation. La distinction entre citation, révision et invalidation existe notamment dans le modèle de provenance W3C PROV-O. 5

L'objectif opérationnel est de pouvoir répondre à une question précise : cette conclusion a été produite à partir de quel état du document ? Une nouvelle version peut imposer une

nouvelle vérification. L'ancienne citation peut rester utile pour comprendre une décision passée, avec son contexte historique clairement visible.

Préparer la reprise sans multiplier les effets

La fraîcheur dépend aussi du redémarrage après une interruption. Un consommateur peut avoir appliqué une mise à jour, puis s'arrêter avant d'enregistrer sa progression. À la reprise, il peut rencontrer de nouveau le même événement.

Notre fiche d'exercice utilise l'événement fictif e-417, qui demande le passage de C-17 à v2. L'oracle de la simulation est explicite : deux tentatives doivent produire un seul effet métier, tout en laissant deux tentatives visibles dans la trace. C'est le résultat attendu de l'exercice, pas une garantie attribuée à MongoDB.

La spécification des change streams rappelle que l'idempotence du traitement relève de l'application. 6 Il faut définir une identité d'événement, une stratégie de déduplication ou une écriture idempotente, puis examiner les frontières entre mise à jour et enregistrement de la progression. Les effets externes nécessitent leur propre traitement.

Un token de reprise ne remplace pas ces contrôles. La reprise dépend aussi de l'historique encore disponible dans l'oplog. 7

Mesurer ce que l'équipe doit pouvoir expliquer

Une revue d'exploitation peut suivre le retard de chaque étape, les changements non traités, les invalidations en échec et les reprises nécessitant une intervention. Les seuils doivent être définis avec les responsables métier plutôt que copiés d'un exemple.

Pour chaque alerte, préciser qui intervient, quelle réponse peut encore être servie et comment revenir à un état vérifié. Il faut aussi conserver assez de traces pour reconstruire le parcours d'une réponse sans exposer inutilement le contenu documentaire.

La question finale de la recette est simple : pouvons-nous montrer pourquoi cette personne a reçu cette version à cet instant ? Une réponse claire rapproche l'ingénierie du besoin métier.

Mettre la chaîne à l'épreuve

Commencez avec un document fictif et une question stable. Faites varier une seule condition à la fois : correction, suppression, révocation, puis reprise du traitement. Pour chaque scénario, écrivez le résultat attendu avant l'exécution et conservez le résultat observé séparément.

Vous obtenez ainsi une revue reproductible de la source jusqu'à la réponse, avec les écarts à corriger et les décisions à prendre.

Pour préparer cette revue avec DATANALYSIS, apportez le schéma de votre pipeline et un exemple de document dont l'actualité compte dans une décision : préparer une revue de fraîcheur du RAG.

Questions & réponses

Pourquoi une réponse RAG peut-elle être périmée alors que la source est à jour ?

L'extraction, l'index ou le cache peuvent encore utiliser une ancienne version. MongoDB Search et Vector Search sont à cohérence éventuelle. Il faut retrouver la version de chaque étape et vérifier si une réponse a été générée ou réutilisée.

Une durée d'expiration du cache suffit-elle à garantir la fraîcheur ?

Une expiration borne la conservation, mais une correction critique ou un changement de droits peut nécessiter une invalidation plus tôt. Le comportement attendu dépend de l'usage et doit être défini puis testé par l'application.

La suppression et la révocation se testent-elles de la même manière ?

Non. La suppression retire le document de la source ; la révocation retire un droit alors que le document peut toujours exister. Les copies, citations et réponses mémorisées doivent être examinées séparément pour chaque scénario.

Que faut-il conserver avec une citation ?

L'identifiant documentaire, sa version, le passage utilisé et la date de consultation. Un lien qui fonctionne ne suffit pas à établir l'actualité du passage ni le droit du destinataire à le lire.

Que prouve l'exercice e-417 avec deux tentatives ?

Il définit un résultat attendu : deux tentatives visibles et un seul effet métier. Il ne prouve pas qu'un système réel respecte déjà cet attendu et n'attribue aucune garantie d'exécution unique à MongoDB.

Par quel test commencer une revue de fraîcheur ?

Choisissez un document fictif et une question stable. Écrivez le résultat attendu, changez une seule condition, puis consignez le résultat observé. Commencez par une correction avant de traiter séparément suppression, révocation et reprise.

Sources

Documentation revue le 2 novembre 2026.

1. MongoDB Search : indexation et cohérence éventuelle
2. MongoDB Vector Search : considérations sur les index
3. MongoDB : événement delete
4. MongoDB : architecture multi-tenant et préfiltrage
5. W3C : PROV-O
6. MongoDB : spécification des change streams et idempotence applicative
7. MongoDB : reprise des change streams