# Guide d'optimisation des performances du classeur SuiteAnalytics

By Houseblend Publié le 6 août 2025 50 min de lecture


# Optimisation de la performance des classeurs SuiteAnalytics pour les grands ensembles de données

## Introduction

SuiteAnalytics Workbook est l'outil d'analyse natif puissant de NetSuite pour l'exploration de données en temps réel, permettant aux utilisateurs de créer des ensembles de données personnalisés, des tableaux croisés dynamiques, des graphiques et des rapports sans écrire de code (Source: annexa.com.au). À mesure que les entreprises accumulent des volumes de données de plus en plus importants, il devient essentiel de s'assurer que les classeurs SuiteAnalytics fonctionnent efficacement avec de grands

ensembles de données. Des classeurs mal optimisés peuvent entraîner des temps de chargement lents, des délais d'attente (timeouts) ou un système surchargé, ce qui a un impact sur les développeurs NetSuite, les analystes commerciaux et les professionnels des données. Ce rapport fournit un guide complet de niveau professionnel pour comprendre l'architecture de SuiteAnalytics Workbook, identifier les goulots d'étranglement courants en matière de performance et appliquer les meilleures pratiques d'optimisation. Il couvre les stratégies d'indexation, de jointures et de filtrage ; les techniques de préparation et de modélisation des données ; des conseils de conception de classeurs pour l'efficacité ; des comparaisons avec les recherches enregistrées et les rapports standard ; des exemples concrets ; des outils de dépannage et de surveillance ; et une liste de contrôle d'optimisation des performances étape par étape. En suivant ces directives, les utilisateurs de NetSuite peuvent exploiter toute la puissance des classeurs SuiteAnalytics sur de grands ensembles de données tout en maintenant des analyses rapides et réactives pour une prise de décision éclairée.

## Comprendre l'architecture des classeurs SuiteAnalytics

SuiteAnalytics Workbook fonctionne sur la source de données d'analyse unifiée de NetSuite, introduite en 2019 pour fournir une vue cohérente, basée sur SQL, des enregistrements NetSuite (Source: docs.oracle.com). Contrairement aux recherches enregistrées et rapports hérités qui exposaient parfois les données différemment, la **Source de Données Analytiques** sous-jacente du classeur assure une dénomination cohérente des champs et une récupération des données à travers les types d'enregistrements (Source: docs.oracle.com). En termes pratiques, cela signifie qu'un classeur accède directement à la même base de données transactionnelle en temps réel qui alimente l'ensemble de l' ERP, mais via un nouveau moteur de requête optimisé. Chaque **Classeur** se compose d'un ou plusieurs **Ensembles de Données** – essentiellement des requêtes définies qui regroupent des champs (colonnes), des critères (filtres) et des jointures à partir d'un ou plusieurs types d'enregistrements (Source: annexa.com.au)(Source: annexa.com.au). Le classeur fournit ensuite des outils de visualisation interactifs (tableaux, tableaux croisés dynamiques, graphiques) basés sur ces ensembles de données pour l'analyse (Source: annexa.com.au).

**Jointures Multi-Niveaux :** L'une des avancées architecturales de SuiteAnalytics Workbook est la prise en charge des jointures multi-niveaux entre les types d'enregistrements, surmontant les limitations des recherches enregistrées traditionnelles (Source: annexa.com.au). En coulisses, lorsque vous ajoutez plusieurs types d'enregistrements à un ensemble de données, le système exécute des jointures de style SQL (spécifiquement, des jointures externes gauches pour chaque type d'enregistrement supplémentaire) en arrière-plan (Source: docs.oracle.com). Cela permet de combiner des données provenant, par exemple, des transactions, des clients et des articles dans un seul ensemble de données. Cependant, par défaut, aucune agrégation n'est effectuée au niveau de la requête de l'ensemble de données – l'ensemble de données est destiné à renvoyer des données brutes et jointes, que le classeur

peut ensuite agréger dans un tableau croisé dynamique ou un graphique (Source: docs.oracle.com) (Source: docs.oracle.com). Il est important de noter que chaque jointure que vous ajoutez inclura **tous** les enregistrements du type d'enregistrement primaire/de base (via une jointure externe gauche), avec les données correspondantes du type joint, ce qui peut entraîner des ensembles de résultats très volumineux si cela n'est pas géré avec soin (Source: docs.oracle.com). (Les sections ultérieures aborderont l'impact de l'ordre et du nombre de jointures sur la performance.)

**Données en Temps Réel et Mise en Cache :** Les classeurs SuiteAnalytics sont par nature en temps réel – à mesure que les utilisateurs filtrent, segmentent ou explorent en profondeur, les requêtes s'exécutent sur des données en direct et se mettent à jour instantanément (Source: annexa.com.au). Pour équilibrer les besoins en temps réel et la performance, NetSuite a introduit un mécanisme de mise en cache. Par défaut (dans les versions récentes de NetSuite), les données pour les visualisations de classeurs (tableaux croisés dynamiques et graphiques) peuvent être mises en cache pour améliorer les temps de chargement, jusqu'à 60 minutes pendant les périodes d'inactivité (Source: docs.oracle.com). En 2021, une fonctionnalité améliorée *« Actualisation Optimisée des Données »* (mise en cache des ensembles de données à la demande) a été ajoutée (Source: netsuite.com). Lorsque la mise en cache est activée pour les ensembles de données, le classeur peut charger les données à partir d'un ensemble de résultats pré-mis en cache (datant d'une heure maximum) au lieu d'interroger la base de données en direct à chaque fois, accélérant considérablement l'ouverture et la manipulation de grands tableaux ou graphiques (Source: netsuite.com). Les utilisateurs ont toujours la possibilité de forcer une actualisation pour obtenir des données en temps réel si nécessaire (Source: netsuite.com). Essentiellement, l'architecture offre un *compromis entre vitesse et fraîcheur* : vous pouvez choisir d'échanger un peu d'immédiateté en temps réel contre une performance plus rapide en utilisant des données mises en cache (Source: netsuite.com). Ce cache est automatiquement activé pour les classeurs dans les versions plus récentes, avec des options dans l'interface utilisateur pour basculer en mode temps réel si nécessaire (Source: netsuite.com).

**Composants du Classeur :** En interne, un classeur SuiteAnalytics peut être considéré comme une collection de composants :

- **Ensembles de Données** – les requêtes définissant les données à récupérer (y compris le type d'enregistrement, les champs, les filtres et les jointures). Chaque ensemble de données est basé sur un type d'enregistrement principal et peut joindre des types d'enregistrements liés supplémentaires.
- **Vues de Tableau** – des sorties de tableau simples des résultats bruts d'un ensemble de données (très similaires à un résultat de recherche enregistrée).
- **Tableaux Croisés Dynamiques** – des résumés multidimensionnels de l'ensemble de données, permettant l'agrégation (SOMME, COMPTE, etc.) sur les lignes et les colonnes pour une analyse plus approfondie.

- **Graphiques** – des représentations visuelles (barres, lignes, secteurs, etc.) basées sur des données brutes ou pivotées.

- **Ensembles de Données Liés** – introduits dans les versions ultérieures, permettent de combiner deux ensembles de données distincts dans une visualisation de classeur via des clés communes (effectuant essentiellement une union SQL de deux requêtes sur des champs partagés) (Source: docs.oracle.com)(Source: docs.oracle.com). Les ensembles de données liés permettent des scénarios tels que la comparaison de deux ensembles de données (par exemple, budget vs. réel) sans avoir à les joindre dans une seule requête. Seuls deux ensembles de données peuvent être liés par visualisation, et la liaison agrège intrinsèquement sur les champs clés communs (union distincte des résultats) (Source: docs.oracle.com)(Source: docs.oracle.com).

Cette architecture – une source de données unifiée avec des requêtes de type SQL et une mise en cache – est conçue pour gérer des analyses de données volumineuses et complexes directement au sein de NetSuite. Cependant, comprendre comment le classeur exécute les requêtes (par exemple, chaque champ ou filtre ajouté affecte le SQL en coulisses) est essentiel pour optimiser la performance, en particulier à mesure que les volumes de données augmentent. Dans les sections suivantes, nous explorerons les goulots d'étranglement courants en matière de performance et les stratégies pratiques pour optimiser les classeurs pour les grands ensembles de données.

## Goulots d'étranglement courants en matière de performance avec les grands ensembles de données

Bien que les classeurs SuiteAnalytics soient conçus pour l'analyse multidimensionnelle, certains scénarios peuvent entraîner des goulots d'étranglement en matière de performance lors du traitement de volumes de données importants. Reconnaître ces goulots d'étranglement est la première étape pour les atténuer :

- **Chargements de Données Non Filtrées :** Le problème le plus courant est de récupérer *trop de données* sans filtres adéquats. Si aucun filtre de critères n'est appliqué à un ensemble de données, il récupérera par défaut tous les enregistrements du type d'enregistrement de base (Source: medium.com). Pour un environnement à forte transaction, cela pourrait signifier des millions de lignes, ce qui sera lent à récupérer et à afficher (voire à atteindre les limites du système). Les grands ensembles de données non filtrés peuvent submerger le navigateur et le serveur, entraînant une réponse lente ou des délais d'attente. Par exemple, un classeur sur l'enregistrement des Transactions sans filtre de date ou de type tentera de charger l'intégralité de la table des transactions – ce qui n'est clairement pas efficace.

- **Jointures Excessives :** Joindre de nombreux types d'enregistrements dans un seul ensemble de données peut dégrader considérablement la performance. Chaque jointure est une jointure externe gauche en coulisses, incluant toutes les lignes d'enregistrement primaires (Source: docs.oracle.com). Avec plusieurs jointures, l'ensemble de données peut avoir besoin de scanner et de combiner plusieurs grandes tables. La documentation de NetSuite avertit explicitement qu'un trop grand nombre de jointures dans un seul ensemble de données de classeur peut entraîner des problèmes de performance (Source: docs.oracle.com)(Source: docs.oracle.com). De plus, joindre des champs provenant de tables avec des relations un-à-plusieurs (par exemple, les lignes de facture) peut multiplier le nombre de lignes de résultats et le volume de données, augmentant les temps de chargement.

- **Filtres Complexes ou Inefficaces :** Tous les filtres ne sont pas égaux. Certaines conditions de filtre peuvent devenir des goulots d'étranglement, en particulier sur les grands champs de texte ou les champs non indexés. Par exemple, l'utilisation d'un filtre `contient` sur un champ de texte force une analyse de texte complète et peut être *très* lente sur de grands ensembles de données (Source: netsuite.folio3.com). Un symptôme réel : une recherche enregistrée ou un classeur utilisant un filtre « Nom contient X » sur une grande liste d'enregistrements peut expirer ou prendre des minutes à s'exécuter (Source: netsuite.folio3.com). De plus, l'utilisation de nombreuses conditions OU (disjonctions logiques) dans les filtres peut empêcher la requête d'utiliser les index et la ralentir (Source: docs.oracle.com).

- **Champs de Formule et Calculs Lourds :** Les classeurs permettent d'ajouter des champs de formule (expressions) et des mesures calculées. Bien que puissantes, ces formules sont calculées pour chaque ligne ou agrégées à la volée. Les formules complexes (en particulier celles impliquant des instructions CASE, des expressions régulières ou des références à plusieurs champs) peuvent entraîner une dégradation des performances lorsqu'elles sont appliquées à des milliers de lignes. Dans le moteur de requête sous-jacent de NetSuite, de nombreuses formules se traduisent par des champs calculés (non indexables), ce qui signifie qu'elles ne peuvent pas tirer parti des index de base de données (Source: docs.oracle.com). Les champs calculés (comme les champs de formule ou certains champs NetSuite tels que `customer.oncredithold` qui est une valeur calculée) ont intrinsèquement un impact sur la performance (Source: docs.oracle.com). Si un classeur repose fortement sur de tels calculs, il peut s'afficher plus lentement, en particulier sur de grands ensembles de données.

- **Récupération de Données Texte Volumineuses ou CLOB :** Les champs qui stockent du texte très volumineux (par exemple, de longues descriptions, des notes ou des champs HTML) sont coûteux à récupérer et à trier. Le moteur d'analyse les classe comme des champs WLONGVARCHAR (texte très long) et ils peuvent ralentir les requêtes (Source: docs.oracle.com). Par exemple, des champs comme la description détaillée d'un article ou un grand champ de texte libre sur les enregistrements

ajouteront une surcharge. De plus, des limitations connues indiquent que le contenu de ces grands champs de texte n'est pas mis en cache et que seuls les 250 premiers caractères sont pris en compte pour le tri (Source: docs.oracle.com). Par conséquent, l'inclusion de champs de texte volumineux dans un ensemble de données doit être évitée sauf si absolument nécessaire.

- **Cardinalité Élevée et Manque de Synthèse :** Afficher des données détaillées brutes pour d'énormes ensembles de données peut être intrinsèquement lent à consommer pour les utilisateurs. Par exemple, lister chaque ligne de transaction individuelle (des centaines de milliers de lignes) dans une vue de tableau sera naturellement plus lent à charger et à faire défiler, comparé à la visualisation d'un résumé de niveau supérieur (comme les totaux par mois ou par client). Les classeurs qui n'exploitent aucune synthèse (pivot ou autre) et tentent d'agir comme des extractions de données pour tous les enregistrements peuvent rencontrer des problèmes de performance dans l'interface utilisateur. Dans de tels cas, même si la requête s'exécute sur le serveur, le navigateur pourrait avoir du mal à afficher et à manipuler un ensemble de résultats aussi volumineux.

- **Concurrence et Requêtes Répétées :** Dans un environnement multi-utilisateur, si de nombreux utilisateurs exécutent le même classeur lourd ou plusieurs classeurs lourds simultanément (par exemple, une période de pointe de rapports de fin de mois), cela peut mettre le système à rude épreuve. Chaque requête de classeur consomme des ressources serveur (CPU, mémoire) et peut potentiellement verrouiller temporairement certains accès aux données. Bien que l'infrastructure cloud de NetSuite puisse gérer beaucoup de choses, il est à noter que des requêtes analytiques concurrentes lourdes pourraient impacter la performance globale ou entraîner des mises en file d'attente.

Il est clair que les grands ensembles de données amplifient toute inefficacité dans la conception des classeurs. En pratique, les utilisateurs ont observé que l'interrogation de volumes énormes en temps réel peut avoir un impact sur la performance, c'est pourquoi l'application de filtres et de synthèses est recommandée (Source: annexa.com.au). Les propres directives de NetSuite suggèrent de réduire les données (par exemple en utilisant des filtres de critères ou des champs de résumé) avant de les visualiser (Source: annexa.com.au). De même, les utilisateurs de classeurs limitent souvent les plages de dates ou divisent les requêtes pour éviter les délais d'attente qui surviennent parfois avec des extractions de données massives (une limitation également observée dans les recherches enregistrées) (Source: netsuite.folio3.com). Les sections suivantes approfondiront les techniques pour remédier à ces goulots d'étranglement – en se concentrant sur la manière d'utiliser l'indexation, les jointures, le filtrage et d'autres stratégies pour optimiser la performance des classeurs sur de grands ensembles de données.

# Stratégies d'indexation, de jointures et de filtrage

L'optimisation des classeurs SuiteAnalytics pour les données volumineuses commence par une conception de requête intelligente – cela signifie tirer parti des index, minimiser les jointures lourdes et filtrer les données efficacement.

**Utiliser les Champs Indexés pour le Filtrage :** Un principe clé des bases de données est de filtrer sur les colonnes indexées chaque fois que possible pour accélérer l'exécution des requêtes. La source de données d'analyse de NetSuite indexe automatiquement certains champs (en particulier les clés primaires et certaines dates) (Source: docs.oracle.com). Par exemple, l'ID interne (`id`) de chaque enregistrement est indexé, et des champs comme `lastmodifieddate` sont également généralement indexés (Source: docs.oracle.com). En les utilisant dans vos critères, vous permettez au système de rechercher directement les sous-ensembles de données pertinents plutôt que de scanner des tables entières. En pratique, cela signifie :

- **Filtres de Date :** Incluez toujours une plage de dates ou un filtre de période (par exemple, transactions des 12 derniers mois, ou du dernier trimestre) plutôt que de récupérer tout l'historique. Étant donné que des dates comme `lastmodifieddate` ou `trandate` peuvent utiliser des index, le filtrage par date réduit considérablement l'ensemble de résultats aux seuls enregistrements récents (Source: docs.oracle.com)(Source: docs.oracle.com). Ceci est particulièrement utile pour l'analyse incrémentale (par exemple, « montrez-moi les nouveaux enregistrements depuis la semaine dernière ») (Source: docs.oracle.com).

- **Filtres par ID/Clé Primaire :** Si vous avez besoin de récupérer un ensemble spécifique d'enregistrements (par exemple, un client particulier ou un ensemble limité d'articles), le filtrage par ID internes ou d'autres champs clés est idéal. Par exemple, un jeu de données pourrait utiliser une condition comme `ID Client = 123` ou une plage d'ID, en tirant parti de la clé primaire indexée.

- **Éviter les Filtres Non Indexés :** Essayez d'éviter les critères qui ne peuvent pas utiliser les index, tels que « contient » ou « ne contient pas » sur du texte, ou l'application de fonctions aux champs dans les filtres (par exemple, filtrer sur `LOWER(nom)` égal à quelque chose force une analyse complète). Si vous avez besoin de correspondances partielles, envisagez plutôt une correspondance « commence par » ou une correspondance spécifique, ou utilisez plusieurs filtres spécifiques plutôt qu'un « contient » général. La différence peut être frappante : un simple « contient » sur du texte dans un jeu de données énorme pourrait s'exécuter des ordres de grandeur plus lentement (Source: netsuite.folio3.com). Si « contient » est inévitable, combinez-le au moins avec d'autres critères de limitation (comme la date ou le type) pour réduire les données qu'il analyse.

**Filtrer Tôt et Spécifiquement :** Le principe directeur est de *récupérer uniquement les données dont vous avez réellement besoin*. Dans le constructeur de jeu de données, ajoutez des filtres de critères le plus tôt possible pour restreindre les données à la source. Par exemple, si vous analysez des commandes client, spécifiez le statut = « Exécuté » ou le type = « Commande Client » et une plage de dates au niveau du jeu de données, afin que seules les transactions pertinentes soient récupérées. Il est bien plus efficace de laisser la base de données renvoyer 5 000 lignes filtrées que de récupérer 500 000 lignes et d'en ignorer la plupart dans l'interface utilisateur. Le classeur s'exécutera plus rapidement et l'expérience utilisateur s'améliorera. De plus, utilisez des **types de filtres cohérents** si vous combinez plusieurs conditions – le mélange de types de données dans les filtres peut entraîner une surcharge de conversion de type (Source: docs.oracle.com). Par exemple, assurez-vous que les filtres de date sont tous comparés en tant que dates (TO_DATE vs TO_DATE), plutôt que de mélanger les fonctions de date et d'horodatage, afin d'éviter des coûts de conversion inutiles (Source: docs.oracle.com).

**Optimiser les Jointures – Les Maintenir Peu Nombreuses et Efficaces :** Les jointures multi-enregistrements sont une fonctionnalité puissante des classeurs, mais elles doivent être utilisées judicieusement. Voici comment optimiser les jointures :

- **Limiter le Nombre de Jointures :** Ne joignez que les types d'enregistrements supplémentaires nécessaires à votre analyse. Chaque jointure introduit une autre table (et potentiellement beaucoup plus de lignes via la jointure gauche). NetSuite autorise un nombre illimité de jointures dans un jeu de données, mais avertit explicitement qu'un trop grand nombre de jointures peut entraîner des problèmes de performance (Source: docs.oracle.com). Si vous vous retrouvez à joindre, par exemple, 5 ou 6 tables dans un seul jeu de données, demandez-vous si toutes sont nécessaires simultanément ou si l'analyse peut être décomposée en étapes.

- **Tirer Parti des Relations Prédéfinies :** Lors de l'ajout de jointures dans le constructeur de jeu de données, utilisez les champs d'enregistrement liés fournis par NetSuite (il s'agit généralement de relations de clé étrangère que le système peut joindre efficacement). Le Catalogue d'Enregistrements peut être utile pour voir quelles jointures sont disponibles pour un type d'enregistrement donné. Évitez de joindre sur des relations non standard ou des clés calculées si possible.

- **Ordre de Jointure et Volume de Données :** Considérez quel enregistrement vous choisissez comme enregistrement principal (racine) dans votre jeu de données. Toutes les lignes de la table de l'enregistrement principal apparaîtront dans les résultats (avec ou sans correspondances des tables jointes) (Source: docs.oracle.com). Ainsi, si vous avez une relation un-à-plusieurs, faire du côté « un » le principal peut parfois réduire la duplication. Par exemple, joindre des transactions (plusieurs) à des clients (un) – si votre principal est le Client, vous obtiendrez une ligne par client avec des informations de transaction agrégées uniquement si vous utilisez des jeux de données liés ou un tableau croisé dynamique, mais dans une jointure de jeu de données brutes, cela listerait de toute

façon une ligne par transaction une fois que vous ajoutez des champs de transaction. En général, basez votre jeu de données sur l'entité ou le type de transaction qui reflète le niveau de granularité d'analyse dont vous avez besoin, et joignez le reste. Si les deux côtés sont volumineux (par exemple, transactions et articles d'inventaire), les joindre produira naturellement un résultat important. Dans de tels cas, voyez si vous pouvez diviser l'analyse (par exemple, analyser les transactions et les articles séparément avec une clé commune comme l'ID d'article en utilisant des jeux de données liés ou des tableaux croisés dynamiques séparés).

- **Éviter les Jointures Dupliquées :** Ne joignez pas le même type d'enregistrement plusieurs fois inutilement dans un même jeu de données. Par exemple, joindre l'enregistrement Article deux fois via des relations différentes rendra la requête plus complexe. Utilisez une seule jointure et récupérez tous les champs nécessaires via cette seule relation si possible. Les meilleures pratiques SuiteQL indiquent spécifiquement d'éviter d'utiliser le même chemin de jointure plusieurs fois dans une même requête (Source: docs.oracle.com).

- **Jeux de Données Liés comme Alternative :** Si votre analyse nécessite vraiment deux grands jeux de données disparates (par exemple, une liste de données budgétaires vs données réelles), envisagez d'utiliser deux jeux de données séparés et de les lier dans le classeur sur une clé commune (Source: docs.oracle.com)(Source: docs.oracle.com). Le lien exécute chaque requête indépendamment, puis fusionne les résultats, ce qui peut parfois être plus performant qu'une seule requête de jointure monstre, surtout si un jeu de données peut être agrégé. Les résultats des jeux de données liés sont comme une union de deux requêtes, affichant des valeurs distinctes sur les champs communs (Source: docs.oracle.com)(Source: docs.oracle.com). Cette approche réduit également intrinsèquement la duplication et peut agréger les données par ces champs clés (Source: docs.oracle.com)(Source: docs.oracle.com). N'oubliez pas que le lien est limité à deux jeux de données à la fois et nécessite un champ commun clair (comme un ID, une date ou un nom) présent dans les deux.

**Utiliser les Critères vs. les Post-filtres :** Appliquez les filtres au niveau du jeu de données (critères) plutôt que de vous fier uniquement aux filtres dans la vue du tableau croisé dynamique ou du graphique. Bien que les filtres de tableau croisé dynamique/de table (dans la visualisation) soient utiles pour le découpage interactif, ils récupèrent d'abord l'ensemble du jeu de données, puis filtrent côté client ou au niveau intermédiaire. Les filtres de critères, en revanche, restreignent les données provenant de la base de données dès le départ, ce qui est bien plus efficace pour la gestion de grandes quantités de données. Par exemple, filtrer un tableau croisé dynamique sur le dernier trimestre après avoir extrait une année complète de données est beaucoup moins efficace que d'avoir un critère de jeu de données pour la date = dernier trimestre dès le départ.

**Traiter par Lots ou Segmenter les Données si Nécessaire :** Si vous devez absolument récupérer un très grand volume de données (peut-être pour une exportation ou une analyse exhaustive), envisagez de segmenter la requête. Ceci est plus pertinent dans les scénarios SuiteQL ou de script, mais conceptuellement, vous pourriez créer plusieurs jeux de données plus petits, chacun filtrant sur une portion de données (par exemple, jeu de données A pour les transactions avec ID interne 1-100k, jeu de données B pour 100k-200k, etc., ou en divisant par année). Bien que cela ne soit pas typiquement fait dans l'interface utilisateur, cette approche évite une requête géante et divise plutôt le travail en morceaux gérables (Source: docs.oracle.com)(Source: docs.oracle.com). C'est une stratégie de dernier recours pour des volumes de données extrêmes, et généralement pas nécessaire si d'autres optimisations sont effectuées.

En résumé, considérez l'indexation, la jointure et le filtrage comme l'aspect « optimisation des requêtes » de l'optimisation des classeurs. En *filtrant tôt en utilisant des champs indexés*, en *limitant et optimisant les jointures*, et en *évitant les modèles de filtre coûteux*, vous permettez au moteur d'analyse NetSuite de faire moins de travail et de se concentrer uniquement sur les données qui comptent vraiment pour votre analyse. Cela constitue une base solide pour la performance, sur laquelle de bonnes pratiques de modélisation et de conception de données (section suivante) peuvent bâtir d'autres améliorations.

## Bonnes Pratiques de Préparation et de Modélisation des Données

Au-delà de l'optimisation des requêtes, la performance peut être considérablement améliorée par la façon dont vous préparez et modélisez vos données pour l'analyse. Ces bonnes pratiques garantissent que vos classeurs sont efficaces par conception :

**Commencer avec le Plus Petit Jeu de Données Nécessaire :** Un conseil fondamental est de **minimiser la taille de votre jeu de données** dès le départ (Source: stockton10.com). Incluez uniquement les champs nécessaires à votre analyse (en évitant la tentation de récupérer « tout » juste au cas où). Chaque champ supplémentaire ajoute une surcharge – il pourrait nécessiter une jointure ou au moins des données supplémentaires à récupérer et à transmettre. De même, n'incluez que les enregistrements pertinents via des critères. Si votre analyse ne concerne que les clients actifs, par exemple, filtrez les clients inactifs dans le jeu de données. En commençant par un jeu de données allégé, vous réduisez le temps de traitement et l'utilisation de la mémoire. Une recommandation d'expert est de « commencer avec le plus petit jeu de données possible et d'ajouter uniquement les champs nécessaires » (Source: stockton10.com).

**Utiliser des Données Récapitulatives ou la Pré-Agrégation :** Demandez-vous si vous avez besoin de données brutes au niveau transactionnel ou si des données récapitulatives suffiraient. Souvent, les questions métier trouvent leur réponse dans des métriques agrégées (ventes par mois, dépenses

moyennes par client, etc.). Si vous avez fréquemment besoin d'un résumé coûteux à calculer à la volée, vous avez plusieurs options :

- **Utiliser les Tableaux Croisés Dynamiques des Classeurs pour les Résumés :** La fonctionnalité de tableau croisé dynamique dans les classeurs est excellente pour agréger de grands ensembles d'enregistrements en résultats récapitulatifs (par exemple, regroupement par périodes, catégories, etc.). De cette façon, même si le jeu de données brut est grand, l'utilisateur interagit avec une vue agrégée beaucoup plus petite.

- **Créer des Jeux de Données Récapitulatifs :** Dans certains cas, vous pourriez pré-résumer les données dans un enregistrement personnalisé ou une recherche enregistrée qui alimente le classeur. Par exemple, un script nocturne ou une recherche enregistrée planifiée pourrait calculer les totaux quotidiens par client dans un enregistrement personnalisé « Résumé des Ventes ». Votre classeur peut alors simplement interroger cet enregistrement récapitulatif (qui contient beaucoup moins de lignes que les transactions brutes). Les conseils d'optimisation NetSuite de Stockton préconisent la création de « jeux de données récapitulatifs pour les métriques fréquemment consultées » afin d'éviter de traiter à plusieurs reprises les mêmes grandes quantités de données (Source: stockton10.com). Essentiellement, si 90 % de votre analyse ne nécessite que des totaux mensuels, ne pas interroger les enregistrements quotidiens à chaque fois – interrogez un résumé mensuel préparé.

- **Mises à Jour Incrémentielles :** Si vous devez travailler avec de très grands jeux de données cumulatifs, concevez votre processus pour qu'il se mette à jour de manière incrémentielle. Par exemple, si vous maintenez un jeu de données de « toutes les commandes à ce jour », envisagez de le diviser par année ou par trimestre, ou de le mettre à jour avec uniquement de nouveaux enregistrements (en utilisant des filtres comme `lastmodifieddate > date de la dernière exécution` (Source: docs.oracle.com)) plutôt que de récupérer l'historique complet à chaque fois.

**Nettoyage et Élagage des Données :** Les grands jeux de données incluent souvent des données qui ne sont pas nécessaires ou qui pourraient être archivées. Travaillez avec les parties prenantes métier pour identifier si les données plus anciennes peuvent être exclues ou archivées dans un système externe. NetSuite ne dispose pas d'un archivage natif des données pour les transactions, mais vous pourriez exporter des enregistrements très anciens vers un référentiel externe (ou un entrepôt de données comme NetSuite Analytics Warehouse) s'ils sont rarement interrogés. De plus, supprimez les champs personnalisés inutilisés ou obsolètes sur les enregistrements – chaque champ personnalisé ajoute une jointure ou une recherche en arrière-plan. Maintenir le modèle de données allégé (en éliminant les champs qui ne sont jamais renseignés ou utilisés) réduit la surcharge (Source: stockton10.com)(Source: stockton10.com). Par exemple, si vous avez un grand jeu de données de Transactions mais un champ de

texte personnalisé qui n'a été utilisé que pour un projet il y a des années et qui est vide pour la plupart des enregistrements, supprimer ce champ de votre jeu de données (ou même du système) peut légèrement améliorer les performances des requêtes en simplifiant la structure des données.

**Modéliser avec des Jointures vs. la Dénormalisation :** Les données NetSuite sont relationnelles, et les classeurs vous permettent de joindre tous les enregistrements liés. Cependant, si vous avez fréquemment besoin de données qui nécessitent plusieurs jointures, demandez-vous si une approche dénormalisée pourrait aider. Par exemple, si chaque analyse de factures nécessite également la région du client et la catégorie d'article, vous pourriez créer des champs de formule ou des champs stockés sur l'enregistrement de transaction pour « Région Client » et « Catégorie d'Article » (renseignés via SuiteScript ou des workflows). De cette façon, le classeur peut obtenir ces valeurs directement à partir du jeu de données de transaction sans joindre les tables Client ou Article. Cela échange un peu de stockage contre de la vitesse de requête (puisque vous évitez les jointures supplémentaires à l'exécution). Cela dit, il faut gérer une telle redondance avec soin pour assurer la cohérence des données. Utilisez cette approche pour les attributs non volatils (comme le secteur d'activité d'un client ou la catégorie d'un article, qui ne changent pas souvent) ou utilisez SuiteScript pour les synchroniser.

**Tirer Parti des Fonctionnalités de la Source de Données Analytiques :** La nouvelle source de données analytiques dans NetSuite apporte des améliorations comme un nommage de champs cohérent, mais notez également ses limitations actuelles. Certains champs calculés présents dans les recherches enregistrées pourraient ne pas être directement disponibles dans les classeurs (Source: docs.oracle.com). Dans de tels cas, vous pourriez avoir besoin de recréer ces calculs sous forme de formules de classeur ou de champs personnalisés. Gardez à l'esprit que ceux-ci seront calculés par ligne, comme discuté. Utilisez également le **Catalogue d'Enregistrements** (via l'API SuiteScript ou l'interface utilisateur NetSuite si disponible) pour voir quels champs sont facilement accessibles pour votre jeu de données afin d'éviter les jointures inutiles (Source: docs.oracle.com). Le Catalogue d'Enregistrements peut montrer la cardinalité (relations un-à-plusieurs vs un-à-un) ce qui vous aide à anticiper si une jointure dupliquera des lignes (Source: docs.oracle.com).

**Envisager NetSuite Analytics Warehouse (NSAW) pour le Big Data :** Si vos besoins d'analyse s'étendent à des volumes de données extrêmement importants (au-delà de ce qui est confortable dans l'ERP en direct) ou à une analyse complexe des tendances historiques, envisagez le déchargement vers NetSuite Analytics Warehouse (un entrepôt de données pour NetSuite alimenté par Oracle) ou un autre outil de BI. SuiteAnalytics Workbook est excellent pour l'analyse opérationnelle en temps réel au sein de NetSuite, mais ce n'est pas un entrepôt de données complet. Le blog Annexa suggère que si les utilisateurs ont besoin d'une analyse avancée ou à long terme qui repousse les limites des classeurs (ou des visualisations plus sophistiquées), l'exportation des données vers des outils comme Tableau, Power

BI ou NSAW pourrait être judicieuse (Source: annexa.com.au). Ce n'est pas une optimisation du classeur en soi, mais une reconnaissance que pour certains scénarios de « big data », l'approche optimale peut être d'utiliser le bon outil (une plateforme d'analyse dédiée) plutôt que de forcer le classeur à tout gérer.

En préparant les données de manière réfléchie – en élaguant ce qui n'est pas nécessaire, en résumant si besoin, et en structurant pour un calcul minimal – vous préparez vos classeurs SuiteAnalytics au succès. En substance, **moins c'est plus** : moins le classeur a de données et de complexité à traiter à l'exécution, plus il sera performant. Ensuite, nous examinerons quelques conseils spécifiques de conception de classeurs pour améliorer davantage les performances, en nous concentrant particulièrement sur la façon dont vous construisez vos vues de classeur (tableaux croisés dynamiques, graphiques, etc.) et utilisez les formules.

## Conseils de Conception des Classeurs (Utilisation des Tableaux Croisés Dynamiques et Efficacité des Formules)

La façon dont vous concevez les visualisations du classeur peut grandement influencer les performances et la convivialité, en particulier sur les grands jeux de données. Voici quelques conseils pratiques sur les tableaux croisés dynamiques, les graphiques et les formules pour assurer l'efficacité :

**Utiliser les Tableaux Croisés Dynamiques pour Agréger de Grandes Quantités de Données :** Les tableaux croisés dynamiques sont un allié de la performance lorsqu'il s'agit de grands ensembles d'enregistrements. Au lieu d'afficher chaque ligne détaillée, un tableau croisé dynamique peut montrer des informations récapitulatives (par exemple, les ventes totales par client par mois) ce qui est non seulement plus rapide à interpréter mais signifie également moins de lignes à afficher. Le moteur de tableau croisé dynamique de NetSuite récupérera toujours les données sous-jacentes, mais il effectue l'agrégation avant de vous présenter les résultats. Cela peut réduire la charge sur le navigateur et le réseau. De plus, les classeurs offrent des fonctionnalités comme les filtres haut/bas dans les tableaux croisés dynamiques – vous pouvez facilement restreindre un tableau croisé dynamique aux « 10 premiers » ou « 10 derniers » valeurs par une mesure (Source: netsuite.com). C'est une excellente optimisation : par exemple, au lieu de visualiser les 5 000 articles, vous pourriez configurer un tableau croisé dynamique pour n'afficher que les 50 premiers articles par chiffre d'affaires, réduisant drastiquement la taille de la sortie tout en se concentrant sur ce qui est important. Ces filtres haut/bas et basés sur les mesures s'exécutent sur les résultats agrégés, ils aident donc à découper les données après agrégation sans requêtes personnalisées supplémentaires (Source: netsuite.com). Utilisez ces fonctionnalités pour simplifier ce que l'utilisateur voit. Afficher un graphique ou un tableau croisé dynamique de données récapitulatives (avec peut-être une option d'exploration) est bien plus rapide que de déverser une table géante.

**Limiter les vues de tableau pour l'exploration des détails :** Une vue de tableau brut est utile pour explorer les détails, mais elle ne devrait pas être l'interface principale pour les grandes quantités de données. Si vous incluez une vue de tableau dans le classeur qui affiche l'ensemble complet des données, sachez que son ouverture tentera de rendre tous les résultats (jusqu'aux limites existantes, potentiellement des dizaines de milliers de lignes). Envisagez d'utiliser les vues de tableau uniquement en combinaison avec des filtres ou pour des ensembles de résultats de taille modérée. Si un tableau doit afficher beaucoup de données, encouragez au moins l'utilisation des filtres intégrés (chaque colonne d'une vue de tableau peut être filtrée dans l'interface utilisateur) ou des fonctionnalités de pagination pour naviguer par blocs.

**Optimiser les formules des classeurs :** Les classeurs vous permettent de définir des champs de formule (dans les jeux de données) et des mesures calculées (dans les tableaux croisés dynamiques). Pour les champs de formule ajoutés à un jeu de données :

- Gardez-les aussi simples que possible. Les opérations arithmétiques ou les instructions CASE de base sont généralement acceptables, mais les manipulations de chaînes complexes ou les formules imbriquées ralentiront le calcul de chaque ligne.
- Si la formule est nécessaire mais lourde, envisagez de la calculer en dehors du classeur (par exemple, via un script planifié qui stocke le résultat dans un champ personnalisé, de sorte que le classeur puisse simplement récupérer cette valeur stockée).
- Soyez attentif aux formules qui reproduisent un champ existant ou qui pourraient être obtenues via une jointure ou des critères différents. Par exemple, au lieu d'une formule pour regrouper les transactions par montant (si la logique est lourde), vous pourriez créer un champ personnalisé ou utiliser un résumé de recherche enregistrée, puis importer ce résultat.

Pour les mesures calculées dans les tableaux croisés dynamiques (comme une formule qui divise un champ sommé par un autre, etc.), celles-ci sont généralement acceptables car elles opèrent sur des données récapitulatives, mais assurez-vous de ne pas faire quelque chose comme sommer une formule de nombreux champs qui auraient pu être sommés dans le jeu de données au préalable. Si vous remarquez qu'un tableau croisé dynamique prend du temps à calculer, essayez de simplifier les expressions ou de calculer les résultats intermédiaires dans le jeu de données.

**Minimiser la complexité des visualisations :** Chaque tableau croisé dynamique ou graphique supplémentaire dans un classeur peut ajouter une surcharge, car il peut déclencher sa propre requête (surtout s'ils utilisent des jeux de données ou des regroupements différents). Bien que les classeurs permettent plusieurs vues (même plusieurs tableaux croisés dynamiques ou graphiques) dans un seul classeur, tenez compte des performances lorsque vous en ajoutez beaucoup. Par exemple, si un classeur contient trois tableaux croisés dynamiques et deux graphiques tous basés sur un grand jeu de données, l'ouverture de ce classeur peut exécuter plusieurs requêtes (pour chaque visualisation). Si les

performances en pâtissent, vous pourriez diviser certaines visualisations en classeurs distincts ou utiliser un seul tableau croisé dynamique avec des filtres sélectionnables par l'utilisateur au lieu de plusieurs tableaux croisés dynamiques. Évitez également les graphiques trop complexes (comme un graphique avec des dizaines de séries ou des milliers de points de données) – non seulement c'est lent, mais c'est aussi difficile à lire. Résumez les données de manière appropriée afin que les graphiques aient un nombre de points gérable.

**Utiliser les portlets de tableau de bord de manière appropriée :** NetSuite permet de placer des graphiques ou des tableaux croisés dynamiques de classeurs sur des tableaux de bord via le portlet Analytics (Source: netsuite.com). C'est une excellente fonctionnalité, mais n'oubliez pas que chaque fois que le tableau de bord se rafraîchit, il chargera ces visualisations de classeur. Ainsi, pour les classeurs avec de grands jeux de données, vous ne voudrez peut-être pas charger automatiquement un tableau croisé dynamique lourd sur le tableau de bord d'accueil pour tous les utilisateurs. Une stratégie consiste à utiliser des sorties de classeur plus petites et de haut niveau sur le tableau de bord (comme un KPI ou un graphique de haut niveau efficace), et à laisser les classeurs détaillés et lourds être ouverts à la demande. Alternativement, utilisez l'option de mise en cache – un portlet de tableau de bord bénéficiera des données mises en cache si disponibles, alors assurez-vous que la mise en cache est activée pour ce classeur afin d'éviter des requêtes constantes.

**Éviter les schémas problématiques connus :** Il existe quelques schémas spécifiques à éviter dans la conception des classeurs :

- Ne dispersez pas trop de champs de texte libre dans les lignes/colonnes des tableaux croisés dynamiques. Par exemple, placer un champ de texte non catégorisé comme ligne dans un tableau croisé dynamique avec des milliers de valeurs uniques est à la fois dénué de sens et lent – il est préférable de catégoriser ou de pré-grouper les données.
- Évitez d'utiliser le tri sur des champs de texte très volumineux ou des résultats de formule lorsque ce n'est pas nécessaire. Le tri peut ajouter une étape supplémentaire ; si vous pouvez vous passer d'un résultat trié (ou si le tri par défaut est acceptable), ignorez-le pour un léger gain de performance (Source: docs.oracle.com).
- Soyez prudent avec les filtres basés sur des mesures dans les tableaux croisés dynamiques impliquant le Min/Max de champs de date ou de texte – à un moment donné, ils n'étaient pas pris en charge (Source: docs.oracle.com), et même s'ils sont ajoutés, ils pourraient être lents. Utilisez des mesures numériques pour le filtrage si possible (par exemple, le nombre ou la somme comme proxy).

**Considération de l'expérience utilisateur :** Pensez toujours à l'expérience de l'utilisateur final. Un classeur peut techniquement traiter 100 000 lignes et les afficher, mais un utilisateur va-t-il vraiment faire défiler tout cela ? Généralement non – ils appliqueraient des filtres supplémentaires ou consulteraient des résumés. Concevez donc l'interface du classeur pour encourager cela : peut-être commencer par une

invite (vous pouvez avoir un filtre que l'utilisateur doit définir, comme une date ou un client, avant que les données n'apparaissent) – de cette façon, ils ne lancent pas accidentellement une requête illimitée. Éduquez les utilisateurs sur le fait qu'ils peuvent et doivent affiner les résultats (peut-être créer plusieurs vues de classeur enregistrées pour les segments courants, plutôt qu'un seul « méga classeur » pour tous les scénarios).

En utilisant intelligemment les tableaux croisés dynamiques pour agréger les données, en maintenant l'efficacité des formules et en contrôlant la complexité des vues de classeur, vous tirez le meilleur parti des capacités de SuiteAnalytics tout en évitant les baisses de performance inutiles. Le classeur sera plus rapide et plus réactif, même lorsqu'il traite de grands jeux de données sous-jacents. Ensuite, nous allons comparer les classeurs SuiteAnalytics avec d'autres outils d'analyse NetSuite (recherches enregistrées et rapports) afin de comprendre les différences de performance et les cas d'utilisation.

## Différences entre les classeurs, les recherches enregistrées et les rapports

NetSuite fournit plusieurs outils pour le reporting et la récupération de données : les recherches enregistrées (Saved Searches), les rapports standard (Report Builder) et les classeurs SuiteAnalytics (SuiteAnalytics Workbooks). Comprendre leurs différences – en particulier en termes de performances et de gestion des grandes quantités de données – peut vous guider pour choisir le bon outil pour chaque tâche ou optimiser en conséquence.

- **Recherches enregistrées :** Une recherche enregistrée est une fonctionnalité de longue date de NetSuite qui permet d'interroger des enregistrements avec des critères et d'afficher les résultats dans une liste (avec des sous-totaux récapitulatifs facultatifs). Les recherches enregistrées sont en temps réel et dynamiques, mais elles ont certaines limitations. En termes de performances, les recherches enregistrées peuvent avoir des difficultés avec de très grands jeux de données – les recherches complexes ou celles qui renvoient des milliers de lignes peuvent expirer ou prendre beaucoup de temps à s'exécuter (Source: netsuite.folio3.com). Par exemple, les utilisateurs constatent souvent qu'une recherche enregistrée avec une large plage de dates ou un filtre de texte `contient` sur une table énorme peut être extrêmement lente, voire nécessiter une planification au lieu d'une visualisation à la demande (Source: netsuite.folio3.com). Les recherches enregistrées renvoient également généralement les données dans un format de tableau plat, ce qui peut être difficile à analyser pour les tendances lorsque les volumes sont élevés (Source: netsuite.folio3.com). Elles peuvent être affichées sur des tableaux de bord, mais souvent avec des résultats mis en cache/planifiés pour atténuer les problèmes de performance (par défaut, une recherche enregistrée de tableau de bord peut être configurée pour se rafraîchir à intervalles, et non en continu en temps réel). En résumé, les recherches enregistrées sont excellentes pour les requêtes rapides et

spécifiques et les exportations, mais pour l'analyse à grande échelle, elles nécessitent souvent des solutions de contournement (comme la réduction des plages de dates, la limitation des colonnes de résultats ou la planification) pour éviter la lenteur (Source: netsuite.folio3.com).

- **Rapports standard (Report Builder) :** Les rapports intégrés de NetSuite (états financiers, rapports de ventes, etc.) utilisent le moteur Report Builder. Ceux-ci sont un peu moins flexibles que les recherches enregistrées (vous choisissez des regroupements de données prédéfinis, et vous ne pouvez souvent pas obtenir à la fois les détails au niveau des transactions et un résumé dans une seule vue). Les rapports pré-agrègent souvent les données (par exemple, un rapport de ventes par client somme les transactions par client et par période). Cela signifie que pour les grands jeux de données, un rapport bien conçu peut être efficace car il récupère des totaux récapitulatifs plutôt que des lignes brutes. Cependant, le Report Builder a ses propres considérations de performance : les rapports personnalisés complexes ou les rapports couvrant de nombreuses années peuvent également prendre du temps à s'exécuter, parfois en arrière-plan plutôt qu'en mode interactif. Les rapports sont généralement destinés à la synthèse et à la sortie imprimée (ils ont une meilleure mise en forme pour l'impression/PDF). Ils ne sont pas aussi interactifs en temps réel – vous exécutez généralement un rapport, puis le visualisez ; si vous modifiez les critères, vous l'exécutez à nouveau. Certains rapports permettent la planification et l'envoi par e-mail. En termes d'architecture, les rapports utilisaient historiquement différentes sources de données sous-jacentes ou des tables pré-résumées, ce qui pouvait être désynchronisé ou incohérent avec les recherches enregistrées pour certains champs (Source: docs.oracle.com). Les classeurs ont été introduits en partie pour unifier cela et offrir plus de cohérence. Ainsi, bien que les rapports puissent gérer de grandes quantités de données sous forme agrégée, ils sont moins adaptés à l'exploration ad-hoc de grands jeux de données détaillés (et vous ne pouvez pas effectuer de jointures multiples ou de formules personnalisées aussi souplement que dans les classeurs).

- **Classeurs SuiteAnalytics :** Les classeurs combinent des aspects des recherches enregistrées et des rapports, ajoutant plus de puissance et de flexibilité. Ils permettent des jointures multi-tables comme un rapport (et au-delà), mais aussi des formules définies par l'utilisateur et des tableaux croisés dynamiques à la volée comme un outil de feuille de calcul externe. Du point de vue des performances, les classeurs exploitent le nouveau moteur d'analyse qui peut gérer des volumes de données plus importants avec plus de fluidité. Comme mentionné précédemment, les classeurs peuvent souvent gérer de grands jeux de données qui feraient expirer les recherches enregistrées (Source: netsuite.folio3.com). Ils offrent également des graphiques interactifs et des fonctions de tableau croisé dynamique que les recherches enregistrées ne peuvent pas faire nativement (Source: netsuite.folio3.com). Un autre avantage est l'actualisation en temps réel : alors que les recherches enregistrées sur un tableau de bord peuvent ne pas toujours afficher des données à la minute près (souvent, elles peuvent être mises en cache ou configurées pour se rafraîchir à certains moments), un classeur peut être actualisé à la demande pour extraire des données en direct chaque fois que

nécessaire (Source: netsuite.folio3.com). De plus, grâce à la mise en cache et à l'optimisation, un classeur bien conçu pourrait remplacer plusieurs recherches enregistrées. En fait, un conseil de pro sur le terrain est qu'un seul classeur SuiteAnalytics, s'il est conçu efficacement, peut remplacer 3 à 5 recherches enregistrées traditionnelles tout en offrant de meilleures performances (Source: stockton10.com). Ceci est possible car le classeur peut combiner plusieurs sources de données et présenter différentes vues (au lieu d'exécuter plusieurs requêtes distinctes).

Pour illustrer la différence : imaginez que vous souhaitez analyser les tendances des ventes ainsi que les niveaux de stock pour les articles. Avec les recherches enregistrées, vous pourriez exécuter une recherche pour les transactions de vente et une autre pour l'inventaire, puis combiner ou comparer manuellement les données (souvent hors ligne dans Excel). Avec un classeur, vous pourriez avoir un jeu de données joignant les ventes et l'inventaire, ou deux jeux de données liés, et voir un tableau croisé dynamique « Ventes vs Inventaire » ensemble, en temps réel. Le moteur du classeur gérerait cette requête multidimensionnelle là où les recherches enregistrées ne peuvent pas le faire facilement, ou nécessiteraient un script personnalisé ou une analyse externe.

**Comparaison des performances :** Il est généralement observé que les requêtes des classeurs SuiteAnalytics s'exécutent plus rapidement ou sont au moins plus évolutives pour les requêtes complexes que les recherches enregistrées équivalentes. L'une des raisons est la source de données analytiques optimisée et potentiellement une meilleure génération de SQL. Par exemple, les requêtes de classeur qui utilisent des index et des jointures appropriées peuvent récupérer les résultats rapidement même sur de grandes tables, tandis qu'une recherche enregistrée pourrait s'étouffer ou être moins efficace en raison de méthodes de requête plus anciennes. Le blog Folio3 a noté que « les classeurs peuvent facilement gérer de grandes quantités de données et fournir des résultats instantanément sur le tableau de bord », tandis que les recherches enregistrées pourraient avoir besoin de limiter les plages de dates ou les colonnes pour récupérer les résultats sans expirer (Source: netsuite.folio3.com). Il a également souligné que les recherches enregistrées chargent les données dans un format tabulaire qui devient difficile à gérer avec de grandes quantités de données, tandis que la capacité du classeur à visualiser (graphiques/tableaux croisés dynamiques) facilite la compréhension des grandes quantités de données (Source: netsuite.folio3.com). De plus, les recherches enregistrées, lorsqu'elles sont ajoutées aux tableaux de bord, peuvent ne pas toujours afficher des données en temps réel à moins d'être actualisées manuellement ou planifiées, mais les graphiques de classeur sur un tableau de bord peuvent être interactifs et en temps réel (avec les considérations de mise en cache mentionnées précédemment) (Source: netsuite.folio3.com).

**Quand utiliser quoi :**

- Si vous avez besoin d'une liste rapide d'enregistrements ou d'une exportation, et que le volume de données est modéré, une recherche enregistrée est souvent suffisante et plus simple.

- Si vous avez besoin d'un rapport formaté (par exemple, un compte de résultat ou quelque chose avec des sous-totaux et une mise en forme d'en-tête/pied de page), les rapports standard sont la solution.
- Si vous avez besoin d'analyser de grandes quantités de données avec flexibilité – par exemple, découper et analyser, créer des métriques personnalisées, joindre plusieurs sources de données et visualiser des tendances – le classeur SuiteAnalytics est le choix supérieur.

Cela dit, les classeurs évoluent continuellement, et apprendre à les utiliser efficacement pourrait nécessiter plus d'expertise que les recherches enregistrées. De plus, tous les champs des recherches enregistrées ne sont pas encore disponibles dans les classeurs (certains champs de cas limites ou valeurs calculées pourraient manquer (Source: docs.oracle.com)). Mais pour les performances sur de grandes quantités de données, le classeur est la solution moderne de NetSuite. Il est conçu pour surmonter de nombreuses limitations des recherches enregistrées, tant fonctionnelles que liées aux performances (Source: annexa.com.au)(Source: annexa.com.au).

En conclusion, pour les grands jeux de données : les **classeurs** sont généralement plus efficaces et performants que les **recherches enregistrées** (grâce à la mise en cache, l'exécution multithread, etc.), et ils permettent une exploration interactive que les **rapports** n'offrent pas. Cependant, chacun a sa place, et souvent ces outils se complètent. Une approche judicieuse consiste à utiliser les recherches enregistrées pour les alertes opérationnelles ou les listes simples, les rapports pour les résumés formels, et les classeurs pour l'analyse approfondie et les tableaux de bord – en tirant parti des forces de chaque outil tout en étant conscient de leurs caractéristiques de performance.

## Études de cas et exemples concrets

Pour ancrer ces meilleures pratiques dans la réalité, examinons quelques scénarios illustrant comment l'optimisation peut améliorer considérablement les performances des classeurs SuiteAnalytics et des tâches d'analyse de données NetSuite associées :

**Étude de cas 1 : Remplacer plusieurs recherches enregistrées par un seul classeur** – Une équipe financière utilisait cinq recherches enregistrées différentes pour analyser divers aspects des comptes clients : factures ouvertes par client, factures en retard, délai moyen de paiement, principaux débiteurs et tendance mensuelle des comptes clients. Ces recherches enregistrées étaient lentes à exécuter (l'une prenait environ 30 secondes et expirait occasionnellement en raison du grand jeu de données de factures) et l'équipe devait souvent exporter les résultats vers Excel pour les combiner et les analyser davantage. La solution a été de créer un classeur SuiteAnalytics complet qui regroupait les données de factures et les données clients dans un seul jeu de données, puis offrait plusieurs vues de tableau croisé dynamique : un tableau croisé dynamique pour les factures ouvertes vs en retard par client (avec des

tranches d'ancienneté), un autre tableau croisé dynamique pour la tendance des comptes clients par mois, et un graphique mettant en évidence les 10 principaux clients par montant impayé. En utilisant la capacité du classeur à joindre les transactions aux clients et à effectuer des agrégations de tableaux croisés dynamiques, ce seul classeur a remplacé les multiples recherches enregistrées (Source: stockton10.com). Les performances se sont améliorées car le gros du travail a été effectué dans une seule requête optimisée avec mise en cache, plutôt que cinq recherches distinctes interrogeant la base de données à plusieurs reprises. L'équipe pouvait actualiser le classeur et obtenir toutes ses informations sur les comptes clients en une seule fois. Selon les experts, un classeur bien conçu peut en effet consolider les fonctionnalités de plusieurs recherches enregistrées tout en offrant de meilleures performances (Source: stockton10.com).

**Étude de cas 2 : Le filtrage améliore les performances de 80 %** – Une entreprise de vente au détail disposait d'un classeur analysant les lignes de commande de vente pour identifier les tendances des ventes de produits. Initialement, le jeu de données était défini sans filtre de date, extrayant toutes les commandes de vente sur 5 ans – environ 500 000 enregistrements de lignes – ce qui prenait plus d'une minute à charger et provoquait souvent le blocage du navigateur. En appliquant une optimisation simple (ajout d'un filtre de critère pour la `Date de transaction` dans la période « Cette année à ce jour » et en permettant à l'utilisateur de choisir d'autres plages de dates prédéfinies via un menu déroulant de filtre), le chargement par défaut a été réduit à environ 50 000 lignes, et le classeur s'est chargé en moins de 10 secondes. Lorsque l'analyse de données plus anciennes était nécessaire, l'utilisateur pouvait sélectionner un filtre de plage de dates différent et attendre un peu plus longtemps, mais l'utilisation quotidienne n'essayait plus de tout charger. Cela correspond aux directives générales selon lesquelles l'interrogation de grands volumes en temps réel nécessite l'utilisation de filtres pour affiner les données (Source: annexa.com.au). L'utilisation d'un champ de date indexé dans le filtre a garanti que la requête n'interrogeait que les partitions de données pertinentes (Source: docs.oracle.com). En conséquence, les performances se sont améliorées d'environ 6 fois (de 60 secondes à 10 secondes dans ce cas) simplement en filtrant tôt.

**Étude de cas 3 : Utilisation de tables récapitulatives pour la vitesse** – Une entreprise manufacturière devait analyser quotidiennement le débit de production par rapport au plan. Initialement, elle avait construit un classeur directement sur les enregistrements d'ordre de fabrication et de transaction de production, mais avec des milliers de transactions de production par jour sur plusieurs années, le classeur était lent lorsqu'il s'agissait d'examiner des périodes plus longues. L'entreprise a décidé de créer un enregistrement personnalisé « Résumé quotidien de production » qui stocke un enregistrement par jour et par ligne de production avec la quantité totale produite et la quantité totale planifiée. Un simple script Map/Reduce s'exécute chaque nuit pour calculer les totaux de la veille et mettre à jour cette table. Le classeur SuiteAnalytics a ensuite été redirigé pour utiliser cet enregistrement récapitulatif comme jeu de données (avec des jointures vers un enregistrement statique « Ligne de production » pour les attributs). Désormais, le jeu de données ne contenait que le nombre d'enregistrements correspondant au

nombre de jours (environ 365 jours * 5 lignes = 1 825 enregistrements par an). L'analyse qui balayait auparavant des dizaines de milliers de lignes de transaction pouvait désormais récupérer quelques centaines d'enregistrements récapitulatifs presque instantanément. Les tableaux croisés dynamiques du classeur (par mois, par ligne de production, etc.) s'exécutaient en 1 à 2 secondes, une amélioration drastique par rapport à l'approche directe précédente. Cela illustre la puissance de la pré-agrégation : en effectuant un peu d'ETL pour préparer les données, les performances du classeur sont devenues ultra-rapides, et il était plus facile pour les gestionnaires de digérer les données résumées. Cette stratégie fait écho à la recommandation de créer des jeux de données récapitulatifs pour les métriques fréquemment nécessaires (Source: stockton10.com).

**Étude de cas 4 : Dépannage d'un classeur lent** – Un administrateur NetSuite a remarqué qu'un classeur d'analyse des ventes particulier était extrêmement lent pour certains utilisateurs, prenant près de 90 secondes à charger. Après enquête, il s'est avéré que le classeur comportait deux jeux de données liés : un pour les devis et un pour les commandes, liés par article et par client pour comparer les taux de conversion. Chaque jeu de données était volumineux, et le lien était effectué sur des champs de texte (nom de l'article et nom du client) plutôt que sur des identifiants, ce qui entraînait des inefficacités (le moteur devait regrouper par des champs de texte de nombreux caractères). L'administrateur a optimisé cela en changeant les clés communes en identifiants internes (qui sont indexés et numériques) et également en réduisant les champs inclus dans chaque jeu de données (en supprimant certaines colonnes inutiles comme les descriptions longues). Le résultat a été une accélération significative – le classeur a commencé à se charger en environ 15 secondes, ce qui, bien que toujours perceptible, représentait une nette amélioration. Cet exemple souligne plusieurs points : l'utilisation de champs clés appropriés pour la liaison (ou la jointure) est importante pour la performance, et l'inclusion de champs de texte volumineux inutiles peut nuire à la vitesse. Il montre également la valeur du dépannage itératif : en examinant comment le classeur était construit et en l'ajustant pour utiliser des champs plus efficaces et moins de points de données, la performance a été ajustée à un niveau acceptable.

Ces cas démontrent qu'en appliquant les principes discutés – filtrage, résumé, utilisation prudente des jointures/liens et minimisation des champs lourds – des améliorations concrètes peuvent être obtenues. Dans un cas, un tableau de bord est passé de 45 secondes à 8 secondes de temps de chargement simplement en optimisant les recherches enregistrées sous-jacentes (Source: stockton10.com), et de même, les classeurs peuvent souvent passer d'inutilisables à rapides avec quelques ajustements. Le thème général est que SuiteAnalytics est puissant mais nécessite une **conception minutieuse pour la performance**, surtout à mesure que les données augmentent (Source: stockton10.com). Dans la section suivante, nous verrons comment dépanner systématiquement les problèmes de performance et surveiller les classeurs, afin que vous puissiez identifier les goulots d'étranglement et vérifier les améliorations.

# Outils de dépannage et de surveillance

Lorsque des problèmes de performance surviennent avec les classeurs SuiteAnalytics, ou pour s'assurer que vos optimisations sont efficaces, il est important de tirer parti des outils disponibles et des approches systématiques pour le dépannage et la surveillance :

**Journal d'audit et journal d'exécution des analyses :** NetSuite fournit un journal d'audit et un journal d'exécution des analyses qui suivent l'utilisation des classeurs et des jeux de données (qui a exécuté quoi, quand, et les modifications apportées). Les utilisateurs disposant de la permission *Administrateur d'analyse* peuvent interroger ces journaux (Source: docs.oracle.com)(Source: docs.oracle.com). Le type d'enregistrement du journal d'exécution (usrexecutionlog) est particulièrement utile car il liste toutes les exécutions de classeurs et de jeux de données des 30 derniers jours, y compris les champs utilisés, les formules, l'utilisateur et l'heure d'exécution (Source: docs.oracle.com)(Source: docs.oracle.com). En créant un simple jeu de données sur ce journal d'exécution, vous pouvez surveiller si certains classeurs sont exécutés extrêmement fréquemment ou prennent beaucoup de temps. *Note :* en raison d'un changement de plateforme, les données du journal d'audit après 2020.1 pourraient ne pas être entièrement disponibles de la même manière (Source: docs.oracle.com). Si le journal d'audit intégré est limité, envisagez une journalisation alternative (comme des hooks de script) pour capturer l'utilisation des classeurs critiques en termes de performance. Néanmoins, la vérification de ces journaux peut indiquer si un utilisateur a exécuté un classeur qui a extrait un nombre extrême de champs ou si un classeur ad-hoc non enregistré est fortement utilisé.

**SuiteApp de gestion des performances d'application (APM) :** La SuiteApp APM de NetSuite est un outil puissant pour identifier les problèmes de performance dans votre compte. Elle se concentre principalement sur les personnalisations (scripts, workflows) et les chargements de pages d'enregistrement, mais elle dispose également d'une section **Analyse des performances de recherche**(Source: docs.oracle.com). Les classeurs peuvent être considérés en interne comme des « recherches » (requêtes), donc si une requête de classeur est particulièrement lente, elle peut apparaître dans les rapports de recherche lente de l'APM. L'APM peut lister les recherches enregistrées ou les requêtes les plus lentes et leurs temps d'exécution, ce qui pourrait détecter un jeu de données de classeur lent. En utilisant le tableau de bord de santé des performances de l'APM (Source: docs.oracle.com), vous pouvez voir si une opération consomme un temps inhabituel. Si un classeur expire ou provoque une surcharge, il pourrait y apparaître pour un diagnostic plus approfondi. Bien que l'APM ne vous dise pas directement « ce classeur a besoin d'un index », il mettra en évidence qu'une certaine requête est une anomalie, vous incitant à approfondir.

**SuiteQL et explication de requête (pour le dépannage avancé) :** Si vous en avez la possibilité, vous pouvez utiliser SuiteQL (le langage de requête de NetSuite similaire à SQL, accessible via SuiteScript ou ODBC) pour émuler ce qu'un classeur fait et l'analyser. Parfois, la conversion de la logique d'un jeu de

données en une requête SuiteQL et son exécution via ODBC avec un « EXPLAIN PLAN » (si disponible) peut éclairer si elle utilise des index ou effectue des analyses complètes de table. Le navigateur NetSuite Connect (navigateur de schéma ODBC) peut également montrer si un champ est indexé. Par exemple, si un classeur est lent, essayez de récupérer un échantillon de ses données avec SuiteQL pour voir si la lenteur est inhérente à la requête ou à quelque chose dans la couche du classeur. Le guide des meilleures pratiques de performance SuiteQL peut éclairer ce processus – par exemple, il vous rappelle que les requêtes utilisant `SELECT *` ou trop de jointures sont plus lentes (Source: docs.oracle.com)(Source: docs.oracle.com), ce qui est parallèle au comportement du classeur. En ajustant la SuiteQL équivalente, vous pouvez appliquer des modifications similaires dans le classeur (comme ne sélectionner que les champs nécessaires, réduire les jointures, etc.).

**Outils de développement du navigateur :** Parfois, le goulot d'étranglement se situe côté client (par exemple, le rendu d'une énorme table dans le navigateur). L'utilisation des outils de développement de votre navigateur web (onglets Réseau et Performance) lors du chargement d'un classeur peut aider. Vous pourriez observer si la majeure partie du temps est consacrée à l'attente du serveur ou si les données arrivent rapidement mais que le navigateur se fige ensuite en les traitant. Si c'est le cas, c'est un signe que vous devriez réduire le volume de données (par exemple, en utilisant un tableau croisé dynamique pour agréger ou en ajoutant une pagination). Si les appels réseau montrent quelque chose comme une très grande charge utile JSON en cours de livraison, cela quantifie la quantité de données que vous extrayez.

**Vérifier les limitations connues et les mises à jour :** Gardez toujours un œil sur les notes de version de NetSuite et les limitations connues des classeurs SuiteAnalytics (Source: docs.oracle.com). Par exemple, nous avons noté précédemment que certaines fonctionnalités comme les filtres de mesure sur les tableaux croisés dynamiques avec des dates Min/Max n'étaient pas prises en charge (Source: docs.oracle.com), ou que les champs de texte très volumineux ne sont pas mis en cache (Source: docs.oracle.com). Si votre scénario de classeur rencontre une limitation (comme des données non mises en cache ou un filtrage incorrect), il peut s'agir d'un problème connu qui sera résolu dans une future version. NetSuite améliore souvent le moteur d'analyse à chaque version (par exemple, 2021.2 a introduit la mise en cache à la demande, 2020 a introduit un meilleur filtrage des tableaux croisés dynamiques, etc.). Si vous suspectez qu'un bug ou une limitation est à l'origine de la lenteur (et non votre conception), recherchez ce problème spécifique dans le Centre d'aide ou les forums d'utilisateurs.

**Retour d'expérience utilisateur et tests :** En plus des outils techniques, n'oubliez pas de recueillir les commentaires des utilisateurs. Les utilisateurs finaux peuvent souvent vous dire « Ça bloque toujours quand je choisis le département X » ou « C'est bien le matin mais lent l'après-midi ». De tels indices peuvent indiquer des problèmes de volume de données ou des problèmes de concurrence. Essayez de reproduire les scénarios lents et testez méthodiquement les changements. Par exemple, si un classeur

est lent pour une valeur de filtre spécifique (peut-être qu'un département a une quantité de données écrasante), envisagez de le diviser en son propre jeu de données ou d'ajouter plus de sous-filtres pour ce cas.

**Optimisation itérative :** Lors du dépannage, modifiez un facteur à la fois et observez les performances. Vous pouvez cloner un classeur, puis effectuer des actions comme supprimer une jointure, supprimer un champ de formule ou ajouter un filtre, et voir quel changement produit la plus grande amélioration. Cela aide à identifier le coupable. Si la suppression d'une jointure particulière accélère considérablement les choses, cette jointure pourrait en être la cause – peut-être que cette table est énorme ou que la condition de jointure n'est pas idéale. Vous pouvez alors vous concentrer sur la manière d'inclure ces données différemment (peut-être via un jeu de données lié ou un résumé).

**Surveillance dans le temps :** Les performances peuvent se dégrader à mesure que les données augmentent. Il est bon de surveiller périodiquement les performances clés des classeurs (trimestriellement, par exemple). Si un classeur qui se chargeait en 5 secondes prend maintenant 15 secondes, recherchez ce qui a changé : est-ce le volume de données (peut-être que le jeu de données a doublé de taille), ou un nouveau champ a-t-il été ajouté ? En détectant ces tendances, vous pouvez vous ajuster de manière proactive. Surveillez également l'utilisation des classeurs par rapport aux recherches enregistrées dans votre compte NetSuite. À mesure que l'adoption de SuiteAnalytics augmente, assurez-vous que votre compte dispose d'un débit suffisant. NetSuite n'impose généralement pas de limites strictes sur les requêtes au-delà des délais d'attente, mais une utilisation extrêmement intensive pourrait inciter à discuter de la nécessité d'un entrepôt de données pour certaines analyses.

En résumé, le dépannage des performances des classeurs implique un mélange d'utilisation des capacités de journalisation/audit de NetSuite, d'outils externes comme APM ou SuiteQL, et de tests systématiques à l'ancienne. En observant et en mesurant, vous pouvez généralement identifier le goulot d'étranglement – qu'il s'agisse d'un filtre non indexé, d'une jointure excessive ou simplement de trop de données – puis appliquer les optimisations discutées précédemment. Les outils de surveillance aident à garantir qu'une fois optimisés, les classeurs restent sains à mesure que l'entreprise et les données évoluent.

## Feuille de route et liste de contrôle pour l'optimisation des performances

L'optimisation des performances des classeurs SuiteAnalytics peut être abordée étape par étape. Utilisez la feuille de route et la liste de contrôle suivantes comme guide pour améliorer et maintenir systématiquement l'efficacité des classeurs, en particulier pour les grands jeux de données :

### 1. Évaluer et étalonner les performances actuelles

- Identifiez les classeurs ou les analyses qui s'exécutent lentement ou qui sont critiques pour vos utilisateurs. Recueillez des métriques de base : Combien de temps prennent-ils pour se charger ? Expirent-ils ? Notez la taille des données impliquées (par exemple, 100 000 transactions pour l'année dernière).

- Vérifiez si les problèmes de performance sont universels ou seulement pour certains filtres ou périodes. Évaluez également l'impact sur l'utilisateur – un classeur lent est-il utilisé quotidiennement sur un tableau de bord (impact élevé) ou un rapport ad-hoc rare (priorité plus faible).

### 2. Examiner la portée et les critères du jeu de données

- **Filtres :** Assurez-vous que chaque jeu de données de classeur dispose de critères appropriés pour limiter les données. Ajoutez des plages de dates, des filtres de statut ou toute autre découpe pour éviter d'extraire des enregistrements inutiles. *Liste de contrôle :* Aucun jeu de données ne doit être complètement non filtré, surtout sur les données transactionnelles (Source: medium.com).

- **Champs indexés :** Dans la mesure du possible, utilisez des filtres sur des champs comme l'ID interne, la date de dernière modification ou d'autres champs indexés (Source: docs.oracle.com). Par exemple, préférez « Date entre X et Y » pour limiter la plage plutôt que l'absence de filtre de date.

- **Supprimer « Contient » si possible :** Remplacez les filtres de texte « contient » par « commence par » ou « est égal à » si vous le pouvez, ou ajoutez des critères supplémentaires pour affiner le domaine de recherche sur lequel le « contient » opère (par exemple, filtrez d'abord par type ou catégorie, puis par « contient » sur le nom).

### 3. Rationaliser les champs et les jointures

- **Minimiser les champs :** Parcourez la liste des champs du jeu de données – supprimez tous les champs qui ne sont pas activement utilisés dans une vue ou une analyse de classeur (Source: stockton10.com). Chaque champ doit justifier sa présence. Si vous l'avez ajouté « juste au cas où » mais ne l'utilisez jamais, supprimez-le.

- **Évaluer les jointures :** Listez tous les types d'enregistrements joints dans le jeu de données. Pour chacun, demandez-vous : ai-je vraiment besoin de données de cet enregistrement ? Certaines jointures peuvent-elles être éliminées en utilisant des champs existants ou des formules sur l'enregistrement principal ? Si une jointure ne fournit qu'un seul champ (par exemple, un nom de classification), envisagez de le remplacer par une formule ou de stocker cette valeur sur l'enregistrement principal pour éviter la jointure.

- **Limiter le nombre de jointures :** Si vous avez plus de, disons, 3 ou 4 jointures, vérifiez leur nécessité. Envisagez d'utiliser des jeux de données liés si vous effectuez effectivement des analyses distinctes combinées (comme l'exemple budget vs réel). N'oubliez pas, trop de jointures = impact potentiel sur les performances (Source: docs.oracle.com).

- **Joindre sur des clés efficaces :** Assurez-vous que les jointures se font sur des clés appropriées (ID internes ou clés fournies par la source de données d'analyse). Évitez de joindre sur des champs de texte ou non indexés.

**4. Optimiser les formules et les expressions**

- **Examiner les champs de formule :** Auditez tous les champs de formule dans le jeu de données. Sont-ils simples ? Pourraient-ils être pré-calculés (via un champ personnalisé ou un script) au lieu d'être calculés à la volée ? Si une formule est complexe et ralentit les choses, essayez de la supprimer à titre de test pour évaluer l'impact.

- **Formules de tableau croisé dynamique :** Vérifiez les mesures calculées dans les tableaux croisés dynamiques. Si vous en avez beaucoup, demandez-vous si toutes sont nécessaires simultanément ou si elles peuvent être simplifiées. Par exemple, au lieu de calculer un ratio dans le tableau croisé dynamique, ajoutez peut-être deux tableaux croisés dynamiques (un pour le numérateur, un pour le dénominateur) si c'est plus clair et potentiellement plus rapide.

- **Calculé vs stocké :** Décidez si certains calculs lourds doivent être déplacés hors du classeur (par exemple, utilisez un processus planifié pour stocker un résultat dans un champ que le classeur pourra ensuite utiliser directement).

**5. Utiliser les options de mise en cache et d'actualisation des données**

- Activez la fonction *Données mises en cache* (Actualisation des données optimisée) pour les jeux de données si elle n'est pas déjà activée (Source: docs.oracle.com)(Source: netsuite.com). Dans NetSuite, allez dans Configuration > Société > Activer les fonctionnalités > Analyse, et assurez-vous que « Mise en cache des jeux de données des classeurs SuiteAnalytics » (ou une fonctionnalité de nom similaire) est activée.

- Dans le classeur, définissez le mode d'actualisation de manière appropriée : utilisez le mode mis en cache pour une utilisation régulière afin d'obtenir un chargement rapide (données datant de moins d'une heure), et indiquez aux utilisateurs comment actualiser en temps réel si nécessaire (Source: netsuite.com). Cela offre flexibilité et performance – normalement un chargement à partir du cache pour la vitesse, mais une actualisation manuelle pour des données à la minute près lorsque cela est requis.

- Vérifiez que la mise en cache fonctionne : après la première exécution, les ouvertures ultérieures du classeur (dans le délai de mise en cache) devraient être nettement plus rapides. Sinon, assurez-vous que le jeu de données est éligible à la mise en cache (certaines limitations comme la présence de champs CLOB peuvent désactiver la mise en cache (Source: docs.oracle.com)).

**6. Tester et mesurer les améliorations**

- Après avoir apporté des modifications (filtres ajoutés, champs supprimés, etc.), testez à nouveau le classeur. Utilisez un chronomètre ou les horodatages du journal d'exécution pour voir la différence. Il est bon d'obtenir des améliorations incrémentales – par exemple, filtrer par année peut réduire le temps de chargement de 60s à 10s. Chaque optimisation peut s'accumuler.

- Assurez-vous que les résultats sont toujours corrects et répondent aux besoins de l'entreprise après les modifications. Parfois, l'ajout d'un filtre peut accidentellement exclure des données nécessaires – vérifiez auprès des utilisateurs que les nouvelles contraintes sont acceptables.

**7. Ajustements du tableau de bord et de l'expérience utilisateur**

- Si un classeur est utilisé sur un tableau de bord ou par de nombreux utilisateurs, envisagez de planifier ou d'échelonner son utilisation. Par exemple, si un classeur lourd n'a pas besoin de données en temps réel, vous pouvez planifier un script pour précharger son cache tôt le matin, afin que les utilisateurs accèdent toujours à un cache chaud.

- Éduquez les utilisateurs : fournissez des directives dans la documentation ou la formation, telles que « ce classeur affiche les données du trimestre en cours par défaut, utilisez le filtre de date pour changer la période » etc., afin qu'ils comprennent comment interagir avec lui efficacement. Encouragez également l'utilisation de l'exploration (cliquer sur les totaux des tableaux croisés dynamiques pour voir les enregistrements sous-jacents) plutôt que d'étendre des listes massives.

- Si certaines analyses sont rarement nécessaires, retirez-les des tableaux de bord par défaut. Proposez-les plutôt comme des classeurs à la demande que les utilisateurs peuvent exécuter au besoin. Réservez l'espace du tableau de bord aux métriques légères et fréquemment utilisées.

**8. Envisagez des approches alternatives si nécessaire**

- Si, après optimisation, une analyse particulière est toujours trop lente (peut-être est-elle intrinsèquement complexe ou à très grande échelle), envisagez des alternatives. Cela peut inclure l'utilisation d'une exportation de recherche enregistrée combinée à Excel/Power BI pour ce cas spécifique, ou l'exploitation de SuiteAnalytics Connect (ODBC) pour extraire les données vers une base de données externe où vous pourrez les indexer et les interroger différemment.

- Alternativement, décomposez le problème : peut-être qu'au lieu d'un seul classeur analysant *tous* les aspects, ayez deux ou trois classeurs spécialisés, chacun optimisé pour une facette des données. Les requêtes plus petites et ciblées sont souvent plus performantes qu'une requête globale.

**9. Surveiller au fil du temps**

- Surveillez les performances des classeurs en permanence. À mesure que les données augmentent, un filtre qui générait 50 000 enregistrements pourrait maintenant en générer 100 000 – il est peut-être temps de resserrer le filtre ou d'archiver les anciennes données. Utilisez les journaux d'exécution ou l'APM pour détecter si un classeur commence à apparaître fréquemment dans les listes de requêtes lentes.

- Sollicitez régulièrement les commentaires des utilisateurs. Les utilisateurs pourraient s'adapter à un classeur lent et ne pas se plaindre, mais leur productivité en souffre – vérifiez donc de manière proactive s'ils trouvent des analyses lentes et résolvez le problème.

**10. Documenter et partager les meilleures pratiques**

- Documentez les optimisations que vous avez appliquées et les résultats (par exemple, « Ajout du filtre X, qui a réduit les lignes de 80 %, améliorant le temps de chargement de 20s à 5s »). Cela contribue à construire la connaissance organisationnelle.

- Partagez des directives générales avec l'équipe : par exemple, une fiche de conseils d'une page sur les meilleures pratiques comme *« Lors de la création d'un classeur, incluez toujours un filtre de date, évitez d'utiliser 'Contient' sur de grands textes, limitez les jointures aux nécessaires, etc. »* Cela garantit que les futurs classeurs sont construits de manière optimale dès le départ, plutôt que d'avoir à les refactoriser plus tard.

Suivre cette liste de contrôle offre une approche structurée de l'optimisation des performances. C'est essentiellement une boucle de **Mesurer → Optimiser → Remesurer**, combinée à des principes de conception proactifs. En intégrant ces étapes dans votre cycle de vie de développement pour l'analyse, vous pouvez prévenir de nombreux problèmes de performance avant qu'ils n'apparaissent et maintenir votre environnement SuiteAnalytics en parfait état de fonctionnement, même à mesure que le volume de vos données augmente.

# Conclusion

L'optimisation des classeurs SuiteAnalytics pour les grands ensembles de données est à la fois un art et une science – elle exige une compréhension de l'architecture sous-jacente et du moteur de données, ainsi que l'application de pratiques réfléchies en matière de modélisation et de conception des données.

Nous avons commencé par explorer comment le classeur SuiteAnalytics exploite la source de données d'analyse unifiée de NetSuite et ses capacités de requête en temps réel (Source: docs.oracle.com) (Source: annexa.com.au), offrant aux utilisateurs des outils puissants pour décortiquer leurs données. Cependant, avec une grande puissance (et de grandes données) vient le besoin d'optimisation : les classeurs non filtrés ou mal structurés peuvent faire face à des goulots d'étranglement de performance tels que des temps de chargement longs ou des délais d'attente.

En identifiant les goulots d'étranglement courants – de la récupération de données illimitée à un trop grand nombre de jointures ou de formules lourdes – nous avons préparé le terrain pour des améliorations ciblées. Nous avons approfondi des stratégies spécifiques : l'utilisation de champs indexés et de filtres efficaces pour réduire drastiquement les données scannées (Source: docs.oracle.com), la limitation des jointures et l'exploitation des liens lorsque cela est approprié pour éviter les requêtes complexes (Source: docs.oracle.com), et la préparation des données (par la synthèse ou la dénormalisation) pour faciliter le travail du classeur (Source: stockton10.com). Nous avons également abordé des conseils de conception tels que l'utilisation de tableaux croisés dynamiques pour l'agrégation, la minimisation de l'utilisation de formules coûteuses et le contrôle de la portée des visualisations pour garantir que, même si l'ensemble de données sous-jacent est volumineux, les informations présentées sont concises et rapides à afficher.

La comparaison des classeurs avec les recherches enregistrées et les rapports a mis en évidence que les classeurs, avec leur moteur moderne et leur mise en cache, sont généralement mieux adaptés à l'analyse à grande échelle, réussissant souvent là où les outils plus anciens peinent (Source: netsuite.folio3.com) (Source: netsuite.folio3.com). Des exemples concrets ont illustré les avantages tangibles de l'optimisation – des temps de chargement plus rapides, des analyses consolidées et des utilisateurs finaux plus satisfaits. Dans un cas, un classeur bien optimisé a non seulement amélioré les performances, mais a également simplifié le flux de travail analytique en remplaçant plusieurs recherches enregistrées (Source: stockton10.com).

Nous avons également souligné l'importance d'une approche systématique du dépannage et de la surveillance. Des outils comme l'Application Performance Management SuiteApp et l'Analytics Audit Trail peuvent fournir des informations sur les requêtes qui nécessitent une attention particulière. Et une liste de contrôle d'optimisation étape par étape garantit que rien n'est laissé au hasard – de l'application de filtres à l'activation de la mise en cache et à l'éducation des utilisateurs.

Pour les développeurs NetSuite, les analystes commerciaux et les professionnels des données, le principal enseignement est que le **classeur SuiteAnalytics peut effectivement gérer efficacement de grands ensembles de données**, mais qu'il doit être utilisé en tenant compte des meilleures pratiques. Tout comme on optimiserait une requête de base de données SQL ou un script, les classeurs bénéficient d'une conception réfléchie : réduire les données à ce qui est pertinent, structurer les requêtes pour

utiliser les forces du système (index, relations) et éviter les pièges connus. Ce faisant, vous libérez tout le potentiel de l'analyse en temps réel dans NetSuite – une analyse interactive et multidimensionnelle qui reste performante même à mesure que vos données commerciales augmentent.

En conclusion, l'optimisation des classeurs SuiteAnalytics est un cheminement continu. À mesure que les volumes de données augmentent et que NetSuite publie de nouvelles améliorations, il est crucial de rester informé des meilleures pratiques. L'effort investi dans l'optimisation se traduit par des informations plus rapides, une utilisation plus efficace des ressources système et, en fin de compte, de meilleures capacités de prise de décision au sein de votre organisation. Grâce aux directives et techniques décrites dans ce rapport, vous pouvez aborder en toute confiance l'analyse de grands ensembles de données dans NetSuite et vous assurer que vos classeurs fournissent des informations opportunes et exploitables sans attente.

**Sources :**

1. Centre d'aide NetSuite – *Documentation sur le classeur SuiteAnalytics et la source de données d'analyse*(Source: docs.oracle.com)(Source: docs.oracle.com) (Source: docs.oracle.com)(Source: docs.oracle.com)
2. Blog Annexa – *« Qu'est-ce que le classeur NetSuite SuiteAnalytics ? »* (Fév 2025) – Aperçu et conseils pour le classeur SuiteAnalytics (Source: annexa.com.au)
3. Blog Oracle NetSuite – *« Aperçu de SuiteAnalytics : la version 2021 Release 2 étend les capacités des classeurs »* – Notes sur les améliorations de performance et la mise en cache (Source: netsuite.com)
4. Blog Oracle NetSuite – *« Plus facile et plus rapide : le classeur SuiteAnalytics offre des améliorations »* (Avril 2020) – Discussion sur la mise en cache vs. l'option de données en temps réel (Source: netsuite.com)
5. Blog Folio3 – *« Comparaison et utilisation des classeurs NetSuite »* (Mars 2023) – Comparaison entre les classeurs, les recherches enregistrées et les rapports (Source: netsuite.folio3.com)(Source: netsuite.folio3.com)
6. Blog Stockton10 – *« 10 conseils d'optimisation de base de données NetSuite éprouvés »* – Conseil n°9 sur l'efficacité du classeur SuiteAnalytics (Source: stockton10.com)(Source: stockton10.com)
7. Aide Oracle – *Meilleures pratiques de performance SuiteQL* – Guide sur l'optimisation des requêtes pertinentes pour les classeurs (Source: docs.oracle.com)(Source: docs.oracle.com)

8. Aide Oracle – *Jointure de types d'enregistrements vs Liaison d'ensembles de données* – Explication des mécanismes de jointure et des considérations de performance (Source: docs.oracle.com) (Source: docs.oracle.com)
9. Blueflame Labs (Medium) – *« Plongée approfondie dans les outils de reporting NetSuite »* (Oct 2024) – Descriptions des recherches enregistrées vs rapports vs SuiteAnalytics (Source: medium.com)
10. Aide NetSuite – *Journal d'audit analytique et journal d'exécution* – Surveillance de l'utilisation des classeurs (autorisation Administrateur analytique) (Source: docs.oracle.com)(Source: docs.oracle.com)

---

Étiquettes: netsuite, classeur-suiteanalytics, optimisation-performances, analyse-donnees, grands-ensembles-donnees, modelisation-donnees, intelligence-affaires

---

## À propos de Houseblend

HouseBlend.io is a specialist NetSuite™ consultancy built for organizations that want ERP and integration projects to accelerate growth—not slow it down. Founded in Montréal in 2019, the firm has become a trusted partner for venture-backed scale-ups and global mid-market enterprises that rely on mission-critical data flows across commerce, finance and operations. HouseBlend's mandate is simple: blend proven business process design with deep technical execution so that clients unlock the full potential of NetSuite while maintaining the agility that first made them successful.

Much of that momentum comes from founder and Managing Partner **Nicolas Bean**, a former Olympic-level athlete and 15-year NetSuite veteran. Bean holds a bachelor's degree in Industrial Engineering from École Polytechnique de Montréal and is triple-certified as a NetSuite ERP Consultant, Administrator and SuiteAnalytics User. His résumé includes four end-to-end corporate turnarounds—two of them M&A exits—giving him a rare ability to translate boardroom strategy into line-of-business realities. Clients frequently cite his direct, "coach-style" leadership for keeping programs on time, on budget and firmly aligned to ROI.

**End-to-end NetSuite delivery.** HouseBlend's core practice covers the full ERP life-cycle: readiness assessments, Solution Design Documents, agile implementation sprints, remediation of legacy customisations, data migration, user training and post-go-live hyper-care. Integration work is conducted by in-house developers certified on SuiteScript, SuiteTalk and RESTlets, ensuring that Shopify, Amazon, Salesforce, HubSpot and more than 100 other SaaS endpoints exchange data with NetSuite in real time. The goal is a single source of truth that collapses manual reconciliation and unlocks enterprise-wide analytics.

**Managed Application Services (MAS).** Once live, clients can outsource day-to-day NetSuite and Celigo® administration to HouseBlend's MAS pod. The service delivers proactive monitoring, release-cycle regression testing, dashboard and report tuning, and 24 × 5 functional support—at a predictable monthly rate. By combining fractional architects with on-demand developers, MAS gives CFOs a scalable alternative to hiring an internal team,

while guaranteeing that new NetSuite features (e.g., OAuth 2.0, AI-driven insights) are adopted securely and on schedule.

**Vertical focus on digital-first brands.** Although HouseBlend is platform-agnostic, the firm has carved out a reputation among e-commerce operators who run omnichannel storefronts on Shopify, BigCommerce or Amazon FBA. For these clients, the team frequently layers Celigo's iPaaS connectors onto NetSuite to automate fulfilment, 3PL inventory sync and revenue recognition—removing the swivel-chair work that throttles scale. An in-house R&D group also publishes "blend recipes" via the company blog, sharing optimisation playbooks and KPIs that cut time-to-value for repeatable use-cases.

**Methodology and culture.** Projects follow a "many touch-points, zero surprises" cadence: weekly executive stand-ups, sprint demos every ten business days, and a living RAID log that keeps risk, assumptions, issues and dependencies transparent to all stakeholders. Internally, consultants pursue ongoing certification tracks and pair with senior architects in a deliberate mentorship model that sustains institutional knowledge. The result is a delivery organisation that can flex from tactical quick-wins to multi-year transformation roadmaps without compromising quality.

**Why it matters.** In a market where ERP initiatives have historically been synonymous with cost overruns, HouseBlend is reframing NetSuite as a growth asset. Whether preparing a VC-backed retailer for its next funding round or rationalising processes after acquisition, the firm delivers the technical depth, operational discipline and business empathy required to make complex integrations invisible—and powerful—for the people who depend on them every day.

---