Université Sorbonne

Master DAC

Département informatique

BDLE

Devoir Maison n°2

16 novembre 2022

par

Oussama Sahli

Encadrant universitaire : Hubert Naacke/ Camelia Constantin/ Mohamed-amine Baazizi/ Bernd Amann

Résumé

L'approche mise en place par l'auteur de l'article, permet de mettre à disposition des contraintes de complétude et de cohérence des dimensions des données ; ainsi que des contraintes statistiques qui peuvent être utilisées pour vérifier la cohérence des dimensions des données. Ces contraintes sont utilisées par les utilisateurs pour définir des contrôles de qualité sur leurs données. "La complétude fait référence au degré auquel une entité comprend les données nécessaires pour décrire un objet du monde réel". Et "la cohérence est définie comme le degré de violation d'un ensemble de règles sémantiques". Par exemple, pour indiquer qu'une colonne de l'ensemble des données doit être complète, on utilise la contrainte 'isComplete', qui permet de vérifier que la colonne ne contient aucune valeurs manquantes. On peut également imposer que les valeurs d'une colonne, dans les données, soient unique. Pour cela, on fait appelle à la contrainte "isUnique", qui permet de vérifier qu'il n'y a pas de doublons dans la colonne. Les contraintes statistiques peuvent servir à effectuer une validation personnalisée sur une statistique particulière des données. Par exemple, la contrainte statistique "hasCountDistinct" peut permettre de comparer le nombre de valeurs distinctes non nulles d'une colonne à un nombre spécifié par l'utilisateur. La contrainte statistique 'hasHistogramValues' peut servir à calculer un histogramme de valeur sur une colonne, pour ensuite comparer le ratio d'une valeur particulière de la colonne à une valeur spécifiée par l'utilisateur. D'autres contraintes sont supportées par l'approche de l'auteur ; elles sont listées dans le tableau 1 de la section 3.1 de l'article. Les contraintes utilisées lors des contrôles définis par l'utilisateur, reposent sur des métriques calculées par le système. Ces métriques permettent de calculer des valeurs (nombre,ratio,type,etc.) concernant la complétude et la cohérence des dimensions des données ; ainsi que des statistiques (maximum, minimum, moyenne, écart-type, histogramme, entropie, nombre de valeurs distinctes,etc.) sur une colonne faisant partie de l'ensemble des données. Ces métriques sont énumérées dans le tableau 2 de la section 3.1 de l'article. Je cite en annexe toutes les contraintes et métriques qui peuvent être utilisées pour qu'un utilisateur puisse définir des contôles sur son ensemble de données.

Les contraintes sont calculées via des ”tests unitaires”. Ces tests permettent de déclarer des contraintes sur les données. De plus, ces contraintes peuvent être combinées avec des fonctions définies par l'utilisateur (UDF), telles que du code personnalisé. C'est à dire que les utilisateurs définissent des contrôles et du code de validation via des contraintes, puis le système traduit ces contraintes en un calcul de métriques. Dans un premier temps, le système commence par établir des vérifications sur les contrôles (et leur contraintes) définis par l'utilisateur. Et dans un second temps, le système utilise les vérifications qu'il a établit sur les contrôles afin de collecter les métriques nécessaires pour évaluer chaque contrôles. Après avoir identifier les métriques nécessaires pour chaque contrôles, la bibliothèque du système choisit, pour chaque métriques, un 'Analyseur' (opérateur physique) capable de calculer cette métrique sur les données. Ces 'Analyseurs' sont ensuite transmis à un 'AnalysisRunner' dans la couche d'exécution du système, dont le but est de programmer l'exécution du calcul de la métrique. Pour chaque métriques, où le re-partitionnement des données n'est pas utile, le programme d'exécution collecte les fonctions d'agrégations requises et les exécute dans une seule requête SparkSQL générée sur les données. Ainsi, cela permet au système d'exécuter efficacement la validation des contraintes en traduisant les vérifications en requêtes d'agrégation. Ensuite, les métriques sont stockées dans une base de données de document (DynamoDB), afin de pouvoir les récupérer plus tard.

Les systèmes de validation de données doivent supportés des scénarios où l'on ingère, en continue, des nouveaux lots d'enregistrements pour un ensemble de données. De ce fait, le système aura besoin d'avoir accès à des métriques mise à jour sur l'ensemble de données, et des métriques sur les nouvelles données. La mise à jour de ces métriques doit se faire sans avoir accès aux données précédentes (elle doit être incrémentale). C'est à dire qu'au lieu de répéter le calcul d'une métrique sur un ensemble de données de plus en plus volumineux, l'auteur fait en sorte que son système met à jour la métrique en ne consommant que le dernier ensemble de données (les données supplémentaire que l'on ajoute), et un 'état' du calcul de la métrique. On peut voir 'l'état' comme une sorte de sauvegarde sur les satistiques calculées sur l'ensemble de données avant l'ajout des données supplémentaires. Initialement, on part d'un 'état' vide ; et au fur et à mesure qu'on rajoute des données supplémentaires, on calcul le nouvel état à partir de l'état précédent et des données supplémentaire qui viennent d'être rajoutées. Pour calculer les nouvelles métriques, après avoir ajouté un nouveau lots de d'enregistrements, il suffit de charger les

statistiques dont on a besoin définit sur l'ensemble de données précédent, et de les mettre à jour à partir des valeurs calculées uniquement sur les nouvelles données ajoutées. Plus haut on a dit que pour calculer une métrique on avait besoin d'un 'Analyseur' ; ici pour les calculs incrémentaux, on parlera "d'Analyseur incrémentaux". La couche d'exécution du système stocke les états des "Analyseurs incrémentaux" dans un système de fichiers distribué S3. Les "Analyseurs incrémentaux" sont des analyseurs "conscients de l'état". C'est à dire qu'ils ont accès aux statistiques définit sur l'ensemble de données précédent. Ce qui leur permet d'effectuer des calculs incrémentaux. Ici, un "Analyseur incrémental" est définit via une classe implémentée en Scala. Cette classe prend en compte le type de métrique à calculer et le type d'état requis. Une autre classe 'StateProvider' s'occupe de récupérer et de faire persister l'état. C'est à l'utilisateur de configurer des fournisseur d'état pour pouvoir gérer l'état, dans le but d'exécuter le calcul incrémental. L'état d'un ensemble de données particulier est stocké dans un répertoire S3. L'appel de la méthode "applyOrUpdateFromPersistedState" permet de calculer la métrique et de faire persister l'état. On doit utiliser deux "StateProvider" afin de calculer la métrique pour le prochain ensemble de données. L'un sert à fournir l'état de l'ancien ensemble de données et, l'autre reçoit la mise à jour calculée à partir de l'ancien état et des données supplémentaire ajoutées. Pour calculer les métriques nécessitant un re-partitionnement des données, on a besoin d'un état composé d'un histogramme sur les données. Lors de la mise à jour de la métrique, on fusionne l'histogramme précédent pour les données courantes avec l'histogramme des données supplémentaires en utilisant une jointure externe sur les colonnes de regroupement. Et on calcule les comptes pour les données supplémentaires et l'ensemble de données. Ensuite, l'analyseur calcule la métrique de l'état en effectuant une agrégation sur l'histogramme.

Annexe

Les contraintes disponibles pour les utilisateurs , afin qu'ils puissent définir des 'tests unitaires' pour leur ensemble de données, sont les suivantes :

Dans un premier temps, on a les contraintes sur la complétude des dimensions :

- isComplete : Cette contrainte permet de vérifier qu'il n'y a pas de valeurs manquantes dans une colonne.
- hasCompleteness : Elle permet d'effectuer une validation personnalisée de la fraction des valeurs manquantes dans une colonne. C'est à dire que par exemple, elle permet de vérifier si le nombre de valeurs non nulles dans une colonne est supérieur ou égale à un certain ratio.

Dans un second temps, on a les contraintes sur la cohérence des dimensions :

- isUnique : Elle permet de vérifier qu'il n'y a pas de doublons dans une colonne.
- hasUniqueness : Elle permet d'effectuer une validation personnalisée du ratio de la valeur unique dans une colonne.
- hasDistinctness : Permet de réaliser une validation personnalisée du ratio de ligne unique dans une colonne.
- isInRange : Permet de déterminer une validation de la fraction des valeurs qui sont dans un intervalle valide.
- hasConsistentType : Réalise une validation de la plus grande fraction des valeurs qui ont le même type.
- isNonNegative : Effectue une validation qui permet de savoir si toutes les valeurs dans une colonne numérique sont non-négatives.
- isLessThan : Permet d'effectuer une validation permmetant de savoir si les valeurs de la première colonne sont toujours inférieures à celles de la deuxième colonne.
- satisfies : Réalise une validation nous indiquant si toutes les lignes correspondent au prédicat.

— satisfiesIf : Détermine une validation nous informant si toutes les lignes correspondant au premier prédicat correspondent également au second prédicat.
— hasPredictabilty : Elle nous informe de la prévisibilité d'une colonne.

Dans un troisième temps, on a les contraintes statistiques , qui peuvent être utilisées pour vérifier la cohérence des dimensions :

— hasSize : Permet d'effectuer une validation sur le nombre d'enregistrements de l'ensemble de données.
— hasTypeConsistency : Permet de réaliser une validation en comparant la fraction maximale des valeurs du même type de données à un nombre définit par l'utilisateur.
— hasCountDistinct : Effectue une validation personnalisée du nombre de valeurs distinctes non nulles dans une colonne.
— hasApproxCountDistinct : Réalise une validation personnalisée du nombre approximatif de valeurs distinctes non nulles.
— hasMin : Détermine une validation personnalisée de la valeur minimale d'une colonne.
— hasMax : Détermine une validation personnalisée de la valeur maximal d'une colonne.
— hasMean : Validation personnalisée de la valeur moyenne d'une colonne
— hasStandardDeviation : Permet de réaliser une validation personnalisée de l'écart-type d'une colonne.
— hasApproxQuantile : Validation personnalisée d'un quantile particulier d'une colonne.
— hasEntropy : Validation personnalisée de l'entropie d'une colonne
— hasMutualInformation : Validation personnalisée de l'information mutuelle d'une paire de colonnes.
— hasHistogramValues : validation personnalisée d'un histogramme de colonne
— hasCorrelation : Validation personnalisée de la corrélation d'une paire de colonnes.

Et enfin, nous avons les contraintes de temps :

— hasNoAnomalies : Validation des anomalies dans les séries temporelles de valeurs métriques.

Comme nous l'avons dit, dans le résumé ci-dessus, les contraintes utilisées lors des contrôles définis par l'utilisateur, reposent sur des métriques calculées par le système.

Parmi ces métriques, premièrement on a les métriques sur la complétude des dimensions :

— Completeness : Fraction des valeurs non manquantes dans une colonne.

Deuxièmement, on a les métriques sur la cohérence des données :

— Size : Le nombre d'enregistrement de l'ensemble de données.
— Compliance : Ratio du nombre d'enregistrements correspondant à un prédicat spécifié par l'utilisateur.
— Uniqueness : Ratio du nombre de valeur unique dans une colonne.
— Distinctness : ratio du nombre de lignes unique dans une colonne.
— ValueRange : Vérification de la plage de valeur pour une colonne.
— DataType : Type de données inféré pour une colonne.
— Predictability : Prévisibilité des valeurs d'une colonne.

Et puis finalement, on a les contraintes statistiques, qui peuvent servir de vérification pour la cohérence des dimensions :

— Minimum : Valeur maximale d'une colonne.
— Maximum : Valeur minimale d'une colonne.
— Mean : Moyenne d'une colonne.
— StandardDeviation : Écart-type de la distribution de valeurs d'une colonne.
— CountDistinct : Nombre de valeurs distinctes dans une colonne.
— ApproxCountDistinct : Nombre de valeurs distinctes dans une colonne estimé en utilisant un croquis de l'hyperlogue.
— ApproxQunatile : Quantile approximatif de la valeur d'une colonne.
— Correlation : Corrélation entre deux colonnes.
— Entropy : Entropie de la distribution des valeurs dans une colonne.
— Histogram : Histogramme d'une colonne éventuellement binnée.
— MutualInformation : Information mutuelle entre deux colonne.