MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université Félix Houphouët-Boigny

TRAVAIL DE GROUPE

Thème :

PIPELINE DU TRAITEMENT D'UNE REQUÊTE SQL :

CAS D'ORACLE ET SQL DEVELOPER

CLASSE :

MASTER 1 DATA-SCIENCE

ENSEIGNANT :

Dr. BAYOMOCK André

Année académique : 2024-2025

SOMMAIRE

INTRODUCTION

Oracle Database est un système de gestion de bases de données relationnelles (SGBDR) largement utilisé, réputé pour sa robustesse, sa scalabilité et ses fonctionnalités avancées. Au cœur de son fonctionnement se trouve le traitement des requêtes SQL (Structured Query Language), un processus complexe mais optimisé qui transforme une simple instruction textuelle en un ensemble d'opérations efficaces pour accéder et manipuler les données. Comprendre ce pipeline de traitement est essentiel pour les développeurs et les administrateurs de bases de données afin d'optimiser les performances des applications et de diagnostiquer les problèmes potentiels.

Cet exposé détaille les différentes étapes du pipeline de traitement des requêtes dans Oracle Database, en mettant un accent particulier sur l'interaction avec le client SQL Developer. SQL Developer, un outil graphique gratuit fourni par Oracle, est couramment utilisé par les développeurs pour interagir avec la base de données, écrire et exécuter des requêtes SQL et PL/SQL, gérer les objets de la base de données, et bien plus encore. Nous examinerons comment une requête SQL, initiée depuis SQL Developer, traverse les différentes phases de traitement au sein du serveur Oracle : de sa réception initiale à la transmission finale des résultats au client.

1. Réception de la requête

Le processus de traitement d'une requête SQL dans Oracle Database commence dès l'instant où un client, tel que SQL Developer, soumet une instruction SQL au serveur de base de données. SQL Developer offre une interface conviviale, notamment via le composant "SQL Worksheet" (Feuille de travail SQL), pour saisir et exécuter des instructions SQL, PL/SQL ou des scripts SQL*Plus.

Lorsqu'un utilisateur tape une requête dans la feuille de travail SQL et clique sur "Exécuter l'instruction" (Execute Statement) ou "Exécuter le script" (Run Script), SQL Developer établit (ou utilise une connexion existante) une session avec le serveur Oracle Database. L'instruction SQL est alors transmise du client SQL Developer au processus serveur dédié ou partagé associé à cette session sur le serveur de base de données. Cette transmission s'effectue via le protocole réseau Oracle Net Services (anciennement SQL*Net).

Du point de vue du serveur Oracle, la réception de l'instruction SQL déclenche ce que la documentation Oracle appelle un "appel d'analyse" (parse call). Cet appel est la première interaction formelle entre l'application (ici, représentée par le processus serveur agissant pour le compte de SQL Developer) et le moteur SQL de la base de données pour cette instruction spécifique. L'objectif de cet appel initial est de préparer l'instruction pour une exécution ultérieure.

Cette étape de réception est fondamentale car elle marque le point d'entrée de la requête dans le moteur de traitement SQL d'Oracle. La manière dont SQL Developer gère l'envoi de ces requêtes (par exemple, l'envoi d'une seule instruction via "Execute Statement" ou d'un bloc de code via "Run Script") influence la façon dont les appels d'analyse sont effectués et potentiellement la performance globale perçue par l'utilisateur.

2. Parsing de la requête

Une fois la requête SQL reçue par le processus serveur Oracle, l'étape suivante et cruciale est le parsing (analyse). Cette phase décompose l'instruction SQL textuelle en une structure de

données interne que les autres composants du moteur de base de données peuvent comprendre et traiter. Comme mentionné dans le Oracle Database SQL Tuning Guide, "The parsing stage involves separating the pieces of a SQL statement into a data structure that other routines can process." 2. Cette étape est initiée par l'appel d'analyse (parse call) effectué par l'application.

Lors de cet appel, Oracle effectue plusieurs vérifications essentielles pour s'assurer de la validité et de la signification de la requête avant même d'envisager son exécution. Ces vérifications se déroulent en plusieurs sous- étapes :

a. Analyse lexicale : Oracle découpe la requête en tokens (mots-clés, identifiants, opérateurs)

b . Vérification Syntaxique (Syntax Check) : Oracle vérifie ensuite si l'instruction respecte les règles grammaticales du langage SQL. Une simple faute de frappe dans un mot-clé (par exemple, FORM au lieu de FROM) ou une structure de clause incorrecte entraînera une erreur à ce stade. Si la syntaxe est invalide, le traitement s'arrête et une erreur (par exemple, ORA-00923: FROM keyword not found where expected) est retournée au client (SQL Developer), qui l'affichera généralement dans le panneau de sortie ou de résultats.

c . Vérification Sémantique (Semantic Check) : Si la syntaxe est correcte, Oracle procède à la vérification sémantique. Cette étape détermine si l'instruction a un sens logique dans le contexte de la base de données.

3. Validation de la requête

Suite à l'analyse syntaxique réussie, la phase de parsing se poursuit avec la validation sémantique de la requête. Bien que la documentation Oracle intègre ces vérifications dans l'étape globale de "Parsing", il est utile de les considérer comme une phase de validation distincte pour bien comprendre les contrôles effectués avant l'optimisation. Cette validation garantit que l'instruction SQL, bien que grammaticalement correcte, est également **logique** et exécutable dans le contexte actuel de la base de données et des privilèges de l'utilisateur.

Les principaux aspects de cette validation sémantique incluent :

1. Validation des Objets et des Colonnes : Oracle vérifie que tous les objets référencés dans la requête (tables, vues, séquences, etc.) existent réellement dans le schéma de la base de

données ou sont accessibles via des synonymes publics ou privés. Il contrôle également que les colonnes spécifiées appartiennent bien aux tables ou vues mentionnées. Une référence à un objet ou une colonne inexistant(e) entraîne une erreur sémantique (par exemple, ORA-00942: table or view does not exist ou ORA-00904: invalid identifier pour une colonne).

2. Résolution des Noms (Name Resolution) : Si des objets ne sont pas qualifiés par le nom de leur schéma, Oracle tente de résoudre leur nom en cherchant d'abord dans le schéma de l'utilisateur courant, puis parmi les synonymes privés, et enfin parmi les synonymes publics.

3. Vérification des Privilèges : C'est une étape de sécurité fondamentale. Oracle s'assure que l'utilisateur qui a soumis la requête via SQL Developer dispose des droits (privilèges système ou objet) nécessaires pour effectuer l'opération demandée sur les objets concernés. Par exemple, pour exécuter un SELECT sur une table, l'utilisateur doit avoir le privilège SELECT sur cette table (directement ou via un rôle). Tenter d'accéder à un objet sans les privilèges appropriés résulte en une erreur ORA-01031: insufficient privileges.

4. Validation des Types de Données et des Fonctions : Oracle vérifie la compatibilité des types de données dans les expressions, les comparaisons et les affectations. Par exemple, tenter d'insérer une chaîne de caractères dans une colonne de type NUMBER sans conversion implicite ou explicite valide peut échouer. L'utilisation de fonctions est également validée (existence, nombre et type des arguments).

Ces vérifications sémantiques sont effectuées en consultant le dictionnaire de données (Data Dictionary), qui contient les métadonnées sur tous les objets de la base de données, les utilisateurs et leurs privilèges. C'est pourquoi l'accès au dictionnaire de données est une composante essentielle de la phase de parsing, en particulier lors d'un hard parse.

Si toutes ces validations réussissent, la requête est considérée comme **sémantiquement valide**. Oracle dispose alors d'une représentation interne de la requête, prête pour l'étape suivante : l'optimisation (dans le cas d'un hard parse) ou directement l'exécution (si un plan valide a été trouvé lors du Shared Pool Check - soft parse).

4. Transformation/Optimisation de la requête

Si la phase de parsing aboutit à un hard parse (c'est-à-dire qu'aucune version pré-analysée et réutilisable de la requête n'a été trouvée dans le Shared Pool), Oracle doit déterminer la manière la plus efficace d'exécuter la requête. C'est le rôle de l'Optimiseur de Requêtes (Query Optimizer), une composante essentielle et sophistiquée du moteur Oracle.

L'optimisation est le processus par lequel Oracle explore différentes manières possibles d'exécuter une instruction SQL et choisit celle qui est jugée la moins coûteuse (généralement en termes de ressources système consommées, comme le CPU et les I/O disque). L'objectif principal est de minimiser le temps d'exécution de la requête.

Cette phase se déroule en plusieurs étapes clés:

1. Transformation de la Requête (Query Transformation) : L'optimiseur commence souvent par réécrire la requête SQL originale en une ou plusieurs requêtes sémantiquement équivalentes mais potentiellement plus faciles à optimiser ou plus performantes. Ces transformations sont basées sur des règles heuristiques et logiques. Quelques exemples courants de transformations incluent :

2. Estimation de la Cardinalité et du Coût (Cardinality and Cost Estimation) : Pour évaluer les différentes stratégies d'exécution possibles, l'optimiseur doit estimer le nombre de lignes (cardinalité) qui seront traitées à chaque étape et le coût associé à chaque opération (accès à une table, jointure, tri, etc.). Ces estimations reposent sur les statistiques de l'optimiseur. Ces statistiques décrivent les données stockées dans les tables et les index (nombre de lignes, nombre de valeurs distinctes, distribution des données, histogrammes, etc.). Des statistiques précises et à jour sont fondamentales pour que l'optimiseur puisse faire des choix éclairés. Si les statistiques sont manquantes, obsolètes ou inexactes, l'optimiseur risque de choisir un plan d'exécution sous-optimal.

3. Génération des Plans d'Exécution (Plan Generation) : En se basant sur la requête (potentiellement transformée) et les estimations de coût dérivées des statistiques, l'optimiseur explore un espace de recherche de plans d'exécution alternatifs. Pour une même requête, il peut exister de nombreuses façons d'accéder aux données et de les combiner :

• **Méthodes d'accès** : Full Table Scan (parcours complet de la ta- ble), Index Scan (accès via un index), Rowid Scan, etc.

• **Ordres de jointure** : L'ordre dans lequel les tables sont jointes.

• **Méthodes de jointure** : Nested Loops Join, Hash Join, Sort Merge Join.

• **Opérations diverses** : Tri (Sort), Groupage (Group By), Filtrage (Filter), etc. L'optimiseur génère différents plans candidats en combinant ces opérations.

4. Sélection du Plan (Plan Selection) : L'optimiseur calcule le coût estimé pour chaque **plan d'exécution** candidat généré. Il sélectionne ensuite le plan ayant le coût estimé le plus bas. Ce plan choisi est celui qui sera utilisé pour l'exécution réelle de la requête.

Le résultat final de la phase d'optimisation (lors d'un hard parse) est un plan d'exécution optimal (selon les estimations de l'optimiseur). Ce plan est une séquence détaillée d'opérations (appelées row sources ou sources de lignes) que le moteur d'exécution devra suivre pour produire le résultat de la requête.

SQL Developer fournit des outils pour visualiser le plan d'exécution choisi par l'optimiseur pour une requête donnée. La fonctionnalité "**Explain Plan**" (accessible via F10 par défaut ou le menu contextuel dans le SQL Worksheet) permet d'afficher le plan d'exécution estimé sans exécuter la requête. Après l'exécution d'une requête, le plan réel utilisé peut souvent être consulté via des outils comme Autotrace ou en interrogeant les vues dynamiques de performance (par exemple, V$SQL_PLAN). L'analyse de ces plans est une compétence clé pour le tuning SQL.

Il est important de noter que l'optimisation ne s'applique généralement pas aux instructions DDL (Data Definition Language, comme CREATE TABLE), sauf si elles contiennent des sous-requêtes DML. Pour les instructions DML (Data Manipulation Language : SELECT, INSERT, UPDATE, DELETE), l'optimisation est une étape fondamentale lors d'un hard parse.

5. Exécution de la requête

Après les phases de parsing, validation et (si nécessaire) optimisation, le moteur Oracle est prêt à exécuter la requête. L'exécution est le processus par lequel le moteur SQL traite activement les données conformément au plan d'exécution généré ou récupéré.

Le plan d'exécution, comme vu précédemment, est une séquence d'opérations appelées sources de lignes (row sources). Chaque source de lignes prend en entrée un ensemble de lignes (potentiellement vide) provenant de l'opération précédente et produit un nouvel ensemble de lignes en sortie, qui sert d'entrée à l'opération suivante. L'ensemble du plan peut être vu comme un arbre (l'execution tree ou parse tree), où les données circulent depuis les opérations de base (accès aux tables/index) vers le haut jusqu'à l'opération finale qui produit le résultat demandé par la requête SELECT (ou effectue les modifications pour INSERT, UPDATE, DELETE).

Le moteur d'exécution SQL (SQL Execution Engine) parcourt l'arbre du plan d'exécution, généralement en commençant par les opérations feuilles (par exemple, un TABLE ACCESS FULL ou INDEX RANGE SCAN) et en remontant. Chaque opération (source de lignes) est exécutée tour à tour :

• **Opérations d'accès** : Lisent les blocs de données depuis les datafiles (via le buffer cache en mémoire) pour les tables et index concernés.

• **Opérations de jointure** : Combinent les lignes provenant de deux sources selon la méthode de jointure spécifiée (Nested Loops, Hash Join, Sort Merge).

• **Opérations de filtrage** : Appliquent les conditions WHERE pour ne conserver que les lignes pertinentes.

• **Opérations de tri et d'agrégation** : Ordonnent les lignes (ORDER BY) ou calculent des agrégats (GROUP BY, SUM, COUNT, etc.).

• Autres opérations : Projections (sélection des colonnes), conversions de type, etc.

Pour les instructions DML qui modifient les données (INSERT, UPDATE, DELETE), l'exécution implique non seulement la lecture des données mais aussi leur modification dans le

buffer cache. Ces modifications génèrent des informations de REDO (journalisation des transactions) et de UNDO (pour la cohérence en lecture et le rollback). Les verrous nécessaires sont acquis sur les lignes ou les blocs concernés pour garantir l'intégrité des données pendant la transaction.

Dans le contexte de SQL Developer, lorsque l'utilisateur exécute une requête SELECT, le moteur d'exécution commence à produire les lignes de résultat. Ces lignes ne sont généralement pas toutes renvoyées immédiatement au client. Elles sont produites et mises à disposition du processus serveur, qui les transmettra ensuite au client par lots (fetch). SQL Developer affiche typiquement les résultats dans l'onglet "Query Result" (Résultat de la requête) de la feuille de travail SQL au fur et à mesure qu'il les reçoit du serveur.

Pour les instructions INSERT, UPDATE, DELETE, ou DDL, SQL Developer affichera généralement un message confirmant le succès de l'opération et le nombre de lignes affectées (pour DML) ou un message indiquant que l'objet a été créé/modifié/supprimé (pour DDL). Ces informations sont également retournées par le serveur Oracle à la fin de l'exécution de l'instruction.

SQL Developer offre également des fonctionnalités pour surveiller l'exécution. Par exemple, l'option "Monitor SQL Status" (disponible dans certaines versions) permet de suivre la progression des instructions SQL en cours d'exécution. De plus, des outils comme "Autotrace" ou l'analyse des plans d'exécution réels (après exécution) peuvent fournir des statis- tiques détaillées sur les ressources consommées à chaque étape du plan d'exécution (nombre de lignes traitées, I/O physiques et logiques, temps CPU), ce qui est essentiel pour identifier les goulots d'étranglement lors du tuning de performances.

L'étape d'exécution est donc le cœur du traitement où le plan est mis en œuvre pour interagir avec les données réelles de la base de données.

6. Conversion des résultats

Une fois que le moteur d'exécution a traité les données conformément au plan d'exécution et a produit les lignes de résultat (pour une requête SELECT), ces données brutes ne sont pas toujours dans le format exact attendu par l'application cliente, comme SQL Developer. Une étape de conversion peut être nécessaire avant la transmission.

Cette conversion intervient principalement pour deux raisons :

1. Conversion de Type de Données Implicite ou Explicite : La re- quête elle-même peut contenir des fonctions de conversion (par exemple, TO_CHAR, TO_NUMBER, TO_DATE) ou Oracle peut effectuer des con- versions implicites si les types de données sources et cibles diffèrent mais sont compatibles. Par exemple, si SQL Developer s'attend à recevoir une date sous forme de chaîne de caractères dans un format spécifique, une conversion peut avoir lieu côté serveur si la colonne source est de type DATE.

2. Adaptation au Protocole Client/Serveur : Les données stockées en interne par Oracle (par exemple, les types NUMBER, DATE, TIMESTAMP, RAW) doivent être converties dans un format standardisé qui peut être transmis via le protocole Oracle Net Services et interprété par le client OCI (Oracle Call Interface) ou JDBC (Java Database Connectivity) utilisé par SQL Developer. Le client spécifie généralement les types de don- nées dans lesquels il souhaite recevoir les résultats lors de la phase de définition de la requête (define phase). Le serveur Oracle effectue alors les conversions nécessaires lors de la récupération des données (fetch phase) pour correspondre aux types demandés par le client.

Par exemple, un type NUMBER Oracle interne peut être converti en une chaîne de caractères, un entier ou un nombre à virgule flottante selon ce que le client a demandé. De même, une DATE Oracle, qui inclut toujours une composante heure, peut être convertie en une chaîne formatée ou un type de date/heure spécifique au langage du client (comme java.sql.Timestamp pour JDBC utilisé par SQL Developer).

Le Oracle Database Concepts guide 8 mentionne implicitement cette conver- sion en décrivant comment les données sont retournées au client : "Oracle Database returns the data to the

application one row or a set of rows at a time.” Le format de ces données retournées est adapté à ce que le client attend.

Il est donc important de comprendre que les données affichées dans SQL Developer peuvent avoir subi une ou plusieurs transformations depuis leur format de stockage brut dans la base de données.

7. Transmission de la réponse au client

La dernière étape du pipeline de traitement d’une requête SELECT est la transmission des lignes de résultat, potentiellement converties, du serveur Oracle vers l’application cliente, en l’occurrence SQL Developer. Pour les instructions DML (INSERT, UPDATE, DELETE) ou DDL, cette étape consiste à transmettre un code de retour, un message de succès ou d’erreur, et éventuellement le nombre de lignes affectées.

Pour les requêtes SELECT :

Le processus de récupération des données par le client est appelé fetch. Une fois que le moteur d’exécution a produit les premières lignes de résul- tat, celles-ci ne sont pas envoyées en une seule fois au client. La transmission se fait généralement par lots (arrays) pour optimiser l’utilisation du réseau.

La taille du lot (fetch size ou array size) peut souvent être configurée au niveau du client ou du driver (par exemple, une préférence dans SQL Developer ou un paramètre JDBC) et a un impact significatif sur les perfor- mances. Une taille de lot plus grande réduit le nombre d’allers-retours réseau nécessaires pour récupérer toutes les données, mais consomme plus de mémoire côté client et serveur pour chaque fetch.

Pour les instructions DML/DDL :

Pour les instructions qui ne retournent pas d’ensemble de lignes, comme

INSERT, UPDATE, DELETE, COMMIT, ROLLBACK, ou les commandes DDL (CREATE,

ALTER, DROP), le serveur Oracle exécute l'instruction complètement. Une fois l'exécution terminée (avec succès ou échec), le serveur renvoie un message de statut au client SQL Developer. Ce message inclut typiquement :

• Un code de retour indiquant le succès ou l'échec.

• En cas d'erreur, le code d'erreur Oracle (par exemple, ORA-xxxxx) et le message d'erreur associé.

• Pour les DML réussis, souvent le nombre de lignes affectées par l'instruction.

SQL Developer affiche ces informations dans le panneau "Script Output" ou via des boîtes de dialogue de confirmation/erreur.

Cette transmission finale clôt le cycle de vie du traitement d'une requête SQL initiée depuis SQL Developer, ramenant le résultat ou le statut de l'opération à l'utilisateur.

CONCLUSION

Le traitement d'une requête SQL dans Oracle Database, depuis sa soumission via un client comme SQL Developer jusqu'à la réception des résultats, est un processus en plusieurs étapes finement orchestré. Chaque phase – réception, parsing (syntaxique, sémantique, shared pool check), validation, optimisation (transformation, estimation, génération de plan), exécution, conversion et transmission – joue un rôle critique dans la garantie de l'exactitude, de la sécurité et de l'efficacité de l'accès aux données. Comprendre ce pipeline permet aux développeurs utilisant SQL Developer d'écrire des requêtes plus performantes (notamment en favorisant les soft parses via les variables de liaison), de diagnostiquer les problèmes en analysant les plans d'exécution, et d'apprécier l'importance des statistiques à jour pour l'optimiseur. Bien que SQL Developer masque une grande partie de cette complexité, une connaissance du fonctionnement interne du serveur Oracle est indispensable pour exploiter pleinement la puissance de la base de données.