420-C42

Langages d'exploitation des bases de données

Partie 12

DDL III

Création, modification et suppression d’objets supplémentaires
requêtes préparées, procédures et fonctions

Résolution d’une instruction

- Il faut d’abord comprendre que pour chaque requête reçue du côté serveur (SGBD), les opérations suivantes sont réalisées :
  1. normalisation
  2. analyse lexicale — découpe la requête en unités lexicales (tokens)
  3. analyse syntaxique — valide que la structure soit conforme à la grammaire SQL
  4. analyse sémantique — valide sens, types et contraintes de la requête
  5. définition d’un plan d’exécution
  6. optimisation du plan d’exécution
  7. exécution
  8. retour du résultat

Résolution d’une instruction

1. normalisation (prétraitement : retrait des espaces et commentaires)
    1. suppression des espaces superflus
    2. élimination des commentaires
    3. normalisation de la casse, des mots clés, des séparateurs, des identifiants, des caractères spéciaux et d’échappement, …
    4. …
2. analyse lexicale (tokenisation ou segmentation en unité lexicale)
    1. tokenisation
    2. identification des *tokens*
    3. préparation et sortie de l’analyse
    4. …

Résolution d’une instruction

3. analyse syntaxique de la requête (validation de la structure de texte)
    1. construction de l’arbre syntaxique
    2. gestion des erreurs syntaxiques
    3. gestion des priorités (opérateurs, parenthésage, …)
    4. optimisations locales
    5. création de la table des symboles
    6. préparation et sortie de l’arbre syntaxique
    7. …

Résolution d’une instruction

4. analyse sémantique de la requête (validation du sens) :
    1. validation de l’existence des objets sollicités
    2. résolution des droits
    3. vérification des types et de la cohérence des domaines
    4. résolution des alias et des noms de variables
    5. vérification des fonctions et procédures
    6. vérification des sous-requêtes et des jointures
    7. validation des conditions booléennes (WHERE)
    8. contrôle d’intégrité référentiel (FK)
    9. contexte transactionnel respecté (isolation de la requête par exemple, …)
    10. …

Résolution d’une instruction

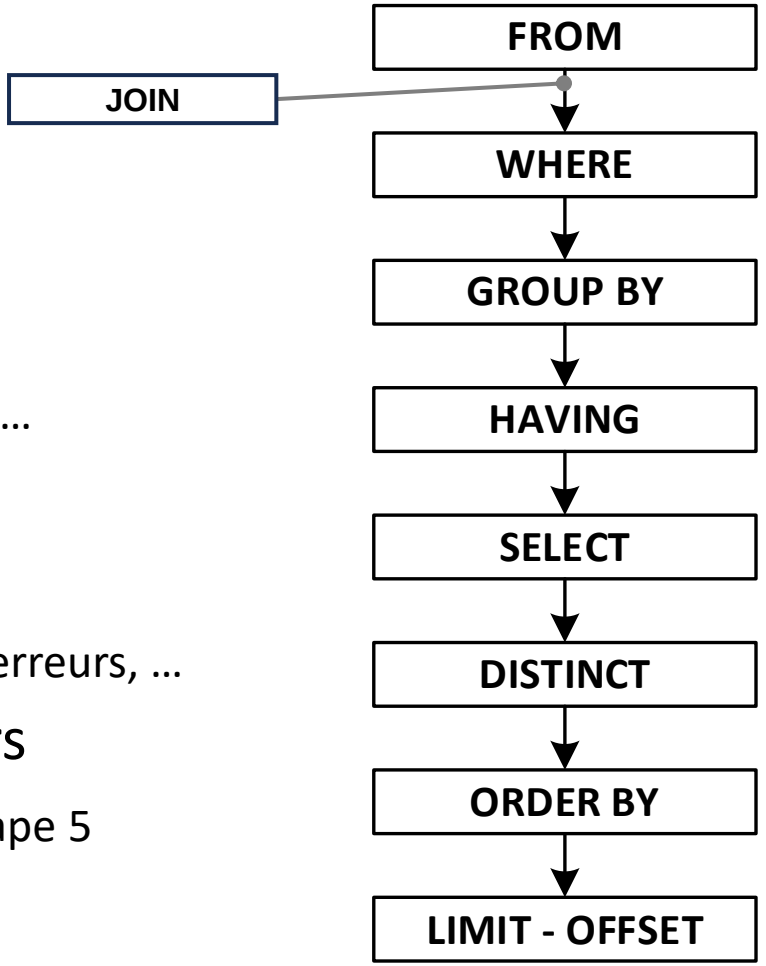
5. définition d’un plan d’exécution
    1. types d’opération à effectuer
    2. ordre des opérations à effectuer
    3. estimation des coûts
    4. prédicats et filtres (conditions à appliquer pour réduire l’ensemble des résultats – optimisation précoce)
    5. utilisation des index
    6. parallélisme
    7. méthodes d’agrégation et regroupement
    8. trie
    9. limite et décalage
    10. …

Résolution d’une instruction

6. optimisation du plan d’exécution
    1. analyse (structures, constituants de la requête, éléments d’information, organisation sur disque, …)
    2. génération de plans candidats
    3. estimation des coûts
    4. estimation des contraintes globales (mémoire RAM et sur disque disponible, …)
    5. optimisations locales
    6. optimisations globales
    7. évaluation parallèle
    8. sélection d’un plan optimal (sélection multicritère)
    9. compilation du plan
    10. mise en cache multiple pour réutilisation
    11. mise en place d’une infrastructure de rétroaction (lors de l’exécution de la requête, des statistiques de performance de la requête seront préservées pour évaluations futures)
    12. …

Résolution d'une instruction

6. Exécution
   1. se rappeler les étapes d'exécution ⇒
7. retour du résultat:
   1. Formatage : transformation des types de données, encodage, …
   2. assemblage : en-têtes, lignes, pagination, …
   3. compression
   4. envoi : tamporisation (« *buffering* »), transmission, gestion des erreurs, …
   5. confirmation de la réception et gestion des erreurs
   6. gestion des métadonnées : les données retournées à l'étape 5



Requête préparée PREPARE

- Une requête préparée (*prepared statement*) permet :
  - une amélioration notable des performances pour des opérations similaires ou récurrentes
  - une simplification de l'écriture des requêtes
- Les deux requêtes suivantes demandent une somme de travail significative et redondante de la part du SGBD même si elles sont structurellement équivalentes :

```
-- insertion de deux nouveaux employés
INSERT INTO employe VALUES ('Bob', 'Montréal', '1990-01-01');
INSERT INTO employe VALUES ('Tom', 'Paris', '1991-02-02');
```

Requête préparée PREPARE

- Une requête préparée permet la mise en place de toutes les opérations liées à la construction de la requête.
- Au final, il ne reste qu'à lier les paramètres variables de la requête et l'exécuter. Ces processus se nomment liaison et exécution (*binding* & *execution*).
- Il va sans dire que les paramètres pouvant être liés sont limités et ne peuvent être l'un de ces éléments : clause, opérateur, objet de la BD, etc.

Requête préparée PREPARE

- Un objet de type requête préparée n'est pas persistant dans la base de données. Il est directement lié à la session de l'utilisateur (connexion à la base de données).
- Les instructions du DDL sont :
  - PREPARE nom_requête_préparée [(type_donnée [, ...])] AS déclaration;
    - Les types des paramètres utiliés sont optionnellement définis.
    - Dans tous les cas, on utilise un substitut (*placeholder*) identifié par $1, $2, $3, … dans la déclaration pour représenter les paramètres de la requête préparée.
  - DEALLOCATE [PREPARE] { nom_requête_préparée | ALL };
- La liaison et l'exécution sont réalisées par :
  - EXECUTE nom_requête_préparée [(paramètre [, ...])]

Requête préparée PREPARE

- Un premier exemple :

```
-- insertion de deux nouveaux employés
PREPARE ins_nouvel_employe_1 AS
    INSERT INTO employe VALUES ($1, $2, $3);
PREPARE ins_nouvel_employe_2(text, text, date) AS
    INSERT INTO employe VALUES ($1, $2, $3);

EXECUTE ins_nouvel_employe_1('Bob', 'Montréal', '1990-01-01');
EXECUTE ins_nouvel_employe_2('Tom', 'Paris', '1991-02-02');

DEALLOCATE ins_nouvel_employe_1;
-- la procédure préparée ins_nouvel_employe_2 sera
-- automatiquement désallouée lorsque l'utilisateur sera déconnecté
```

Requête préparée PREPARE

- Un deuxième exemple :

```
-- insertion d'un nouvel employé avec son département
-- la solution est plus performante et plus facile à utiliser!
PREPARE ins_nouvel_employe_3 AS
    INSERT INTO employe(id, nom, ville, date, departement)
        VALUES (DEFAULT, $1, $2, $3,
                        (SELECT id FROM departement WHERE nom = $4));

EXECUTE ins_nouvel_employe_3('IA', 'Laval', '1992-03-03', 'Ventes');
```

Procédure SQL et fonction SQL

Sommaire

- Le langage SQL rend possible la création de procédures et fonctions.
- Avant de les présenter, il est important de savoir :
  - qu'il existe plusieurs types de procédures et fonctions :
    - SQL : c'est le sujet de ces diapositives
    - PL/pgSQL : que nous verrons plus loin
  - avec PostgreSQL, il est possible de substituer les apostrophes pour délimiter une chaîne de caractères :
    - on utilise à la place des « *dollar-quoted* » formant un bloc personnalisé : ***xyz*** *une chaîne de caractère* ***xyz***
    - où ***xyz*** est défini par l'utilisateur
  - avec PostgreSQL, la surcharge est possible – attention, si vous désirez modifier une procédure ou une fonction, vous devez d'abord la détruire sinon vous risquez de créer une surcharge - crée parfois une situation difficile à déboguer.

Procédure SQL et fonction SQL

Sommaire

- Similarités :
  - Contrairement aux requêtes préparées, les procédures et les fonctions sont persistantes dans la base de données.
  - Comme pour les requêtes préparées, les optimisations mentionnées sont réalisées lorsqu'elles s'appliquent (accroissement de la performance).
  - Dans les deux cas, elles permettent de créer un regroupement d'opérations paramétrables comme le ferait une fonction dans n'importe quel autre langage de programmation.
- Différences :
  - Une procédure ne retourne rien directement et doit être appelé explicitement.
  - Une fonction SQL doit retourner une valeur et peut être appelée dans n'importe quelle requête comme n'importe quelle fonction. Elle peut aussi être appelée explicitement.

Procédure SQL DDL – PROCEDURE

- Les instructions simplifiées du DDL sont :

CREATE [OR REPLACE] PROCEDURE *nom_procedure* ([*liste_arguments*])
  { LANGUAGE *nom_langage* }
  AS 'definition_de_la_procedure';
-- la liste d'arguments est une suite : [*mode*] [*nom*] *type* [{ DEFAULT | = } *valeur*]
-- où *mode* est IN, INOUT ou VARIADIC (IN par défaut)
-- *nom* est le nom de la variable du paramètre
-- *type* est le type de la variable - explicitement ou relatif : *table*.*colonne*%TYPE
-- *valeur* est la valeur par défaut
-- *nom_langage* est SQL ou PLPGSQL

DROP PROCEDURE [IF EXISTS] *nom_procedure* [CASCADE | RESTRICT];

- On utilise la clause CALL pour exécuter une procédure.

Procédure SQL

Exemple

- Un exemple :

```
CREATE OR REPLACE PROCEDURE ajout_employe_nouveau_patron(
        nas_emp employee.nas%TYPE, -- < de loin préférable!
        nom_emp VARCHAR(32),
        prenom_emp VARCHAR(32),
        nom_departement department.name%TYPE)
  LANGUAGE SQL
  AS $$
    INSERT INTO employe(nas, nom, prenom, id_dep)
        VALUES(nas_emp, nom_emp, prenom_emp,
                    (SELECT id FROM departement WHERE nom = nom_departement));
    UPDATE departement
         SET responsable = (SELECT id FROM employe WHERE nas = nas_emp);
         WHERE (SELECT id FROM departement WHERE nom = nom_departement);
  $$;

CALL ajout_employe_nouveau_patron(123, 'Dupuis', 'Lancelot', 'Ventes');
```

Fonction SQL DDL – FUNCTION

- Les instructions simplifiées du DDL sont :

CREATE [OR REPLACE] FUNCTION *nom_fonction* ([*liste_arguments*])
RETURNS *type_retour*
{ LANGUAGE *nom_langage* }
{ CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT }
AS 'definition_de_la_procedure';
-- la liste d'arguments est une suite : [*mode*] [*nom*] *type* [{ DEFAULT | = } *valeur*]
-- où *mode* est IN, INOUT ou VARIADIC (IN par défaut)
-- *nom* est le nom de la variable du paramètre
-- *type* est le type de la variable
-- *valeur* est la valeur par défaut
-- *nom_langage* est SQL ou PLPGSQL
-- *type_retour* est le type de retour de la fonction

DROP FUNCTION [IF EXISTS] *nom_fonction* [CASCADE | RESTRICT];

- On exécute la fonction comme n'importe quelle autre. Il est aussi possible d'appeler la fonction sans SELECT grâce à la clause PERFORM.

Fonction SQL

Exemple

- Un exemple :

```
CREATE OR REPLACE FUNCTION id_departement(
            nom_departement departement.nom%TYPE)
            RETURNS INT
  LANGUAGE SQL
  AS $$
    SELECT id FROM departement WHERE nom = nom_departement;
  $$;

SELECT id_departement('Ventes');
```

Fonction SQL

Exemple

- Un autre exemple :

```
CREATE OR REPLACE FUNCTION dernier_id_employe()
            RETURNS INT
  LANGUAGE SQL
  AS $$
    SELECT id FROM employe ORDER BY id DESC LIMIT 1;
  $$;

SELECT dernier_id_employe();
```

Autres objets

- Tel que mentionné au début de ces diapositives, il existe plusieurs autres types d'objets qu'il est intéressant de connaître. Soyez curieux et explorer!
- Nous verrons dans la section PL/SQL les :
  - extensions des fonctions et des procédures
  - déclencheurs