tellora

Administration

Base de données

PostgreSQL

Objets et concepts

Objets et concepts

- Base de données (rappel)
  - Une base de données est un conteneur de schémas et indirectement de tables et d'index et autres objets de schéma
  - A la connexion d'une application dans une base de données seuls les objets de cette base de données sont visibles
    - Les objets n'appartenant pas à cette base de données ne sont pas visibles
    - Il est important de répartir correctement les applications dans les bases de données en utilisant la notion de schéma

Objets et concepts

- Schéma
  - dans une base de données plusieurs schémas existent à sa création
    - Pg_catalog, contient les tables et vues systèmes qui permettent de stocker des informations relatives à son fonctionnement
    - Public, schéma initialement créé et considéré comme le schéma par défaut
    - Information_schema, contient les informations sur les objets de la base de données courante,
      - informations redondantes avec pg_catalog
  - Le schéma est une notion logique qui permet de regrouper les objets d'une base de donnés au niveau logique
    - Plusieurs schéma peuvent avoir des objets de même nom
    - Plusieurs applications peuvent accéder à la même base de données mais ne pas avoir besoin des mêmes objets
    - Les schémas sont un moyen pratique d'organiser les objets en fonction des applications qui les utilisent
    - Les objets peuvent aussi être regroupés en fonction des habilitations
      - Ainsi on pourra mettre en place les autorisations en fonction du schéma et non de chaque objet

Objets et concepts

- Schéma
  - L'accès à un objet se fait en préfixant l'objet par le nom du schéma :
  - Exemple :
    - Select nom_client from compta.client ;
  - Si vous souhaitez ne pas préfixer les objets par le schéma il faudra que l'objet fasse parti des schémas indiqués par le paramètre « search_path » et que le schéma existe !
  - D'autres schémas peuvent apparaitre et disparaitre en fonction de l'activité du serveur
    - Par exemple une session qui utilise un objet temporaire se verra attribuer un schéma temporaire appelé « pg_tem_id »
      - ID est un numéro unique spécifique à la session

Objets et concepts

- L'ordre CREATE SCHEMA
  - CREATE SCHEMA *nom_schéma* [AUTHORIZATION *spécification_rôle*] [*élément_schéma* [...
  -]]
  - CREATE SCHEMA AUTHORIZATION *spécification_rôle* [*élément_schéma* [...]]
  - CREATE SCHEMA IF NOT EXISTS *nom_schéma* [AUTHORIZATION *spécification_rôle*]
  - CREATE SCHEMA IF NOT EXISTS AUTHORIZATION *spécification_rôle*
  - où *spécification_rôle* peut valoir :
  - [GROUP] *user_name*
  - | CURRENT_USER
  - | SESSION_USER ;

Objets et concepts

- Catalog PostgreSQL
  - Select count(*) from pg_catalog.pg_namespace;
    - Affiche le nombre de schémas disponibles
  - La table pg_user contient les informations sur les utilisateurs et est visible publiquement
    - Select count(*) from pg_user;
    - Affiche le nombre d'utilisateurs créés
  - La table pg_shadow, contient en plus les mots de passe des utilisateurs, elle n'est accessible que par les « superusers »

Objets et concepts

- Table
  - Une table permet de stocker les données de l'utilisateur
    - Taille maximum de 32 Teras
  - Le choix des types de données dépend de la conception
  - Les contraintes sur les colonnes permettent de déclarer une colonne comme :
    - clé primaire
    - Clé étrangère
    - Unique
    - Not null
  - Dans le cas des contraintes clé primaires et unique PostgreSQL crée automatiquement un index
    - Il est possible de choisir le tablespace dans lequel créer l'index avec l'option
      - USING INDEX TABLESPACE

Objets et concepts

- L'ordre CREATE TABLE
  - CREATE [[GLOBAL | LOCAL] { TEMPORARY | TEMP } | UNLOGGED] TABLE [IF NOT EXISTS]
  - *nom_table* ([
  - { *définition des colonnes*
  - *Définition des contraintes }*
  - *])*
  - [INHERITS (*table_parent* [, ...])]
  - [WITH (*parametre_stockage* [= *valeur*] [, ...]) | WITH OIDS | WITHOUT OIDS]
  - [ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP }]
  - [TABLESPACE *nom_tablespace*] ;

Objets et concepts

- Table (suite)
  - Objet permanent par défaut dont les modifications sont enregistrées dans les journaux de transaction
    - Il est possible de créer une table temporaire avec la clause TEMPORARY
      - Une table temporaire n'est visible que de la session qui l'a créée
      - Elle est aussi enregistrée dans les journaux de transaction
      - Elle est automatiquement supprimée à la fin de la session
      - Utilise un cache spécifique à la session dont la taille dépend du paramètre TEMP_BUFFER
      - Il est intéressant de leur allouer un ou plusieurs tablespaces en cas mémoire insuffisante, dans ce cas ils seront utilisé les uns après les autres
        - » Ces tablespaces dépendent de la valeur du paramètre TEMP_TABLESPACES

Objets et concepts

- Tables (suite)
  - Son comportement varie en fonction de la clause ON COMMIT quand elle est créée dans une transaction explicite
    - ON COMMIT PRESERVE ROWS
      - Par défaut
      - Conserve le contenu de la table après la transaction
    - ON COMMIT DELETE ROWS
      - Supprime le contenu de la table en fin de transaction
    - ON COMMIT DROP
      - Supprime la table en fin de transaction

Objets et concepts

- Tables (suite)
  - Dans la définition des tables la clause UNLOGGED permet de créer des tables non journalisées,
    - Ont une définition permanente et globale
    - Elle peuvent être utilisées dans toutes les sessions
    - Les informations ne sont pas enregistrées dans les journaux de transaction, aussi en cas de « crach » elles seront perdues
  - La clause FOREIGN TABLE permet de définir une table externe
    - La structure est enregistrée en local mais les données sont externes à la base
    - L'accès aux données se fait via un connecteur

Objets et concepts

- Tables et notion d'héritage
  - Il est possible de lier des tables entre elles avec une notion d'héritage en utilisant la clause INHERIT
  - Ce qui permet à une table fille d'hériter de la définition des colonnes de la table mère et d'avoir ses propres colonnes en plus
  - Ainsi quand une ligne est insérée dans la table fille, elle apparait aussi dans la table mère
    - Seules les colonnes propres à la table fille sont stockées dans la table fille
  - Cette fonctionnalité remplace le partitionnement qui n'est pas possible en natif dans PostgreSQL

Objets et concepts

- Tables et stockage
  - Par défaut les blocs de table sont remplis le plus possible
  - Taille d'un bloc = 8K
    - La clause FILLFACTOR permet de paramétrer le remplissage des blocs
      - Alter table client set (fillfactor=80) ;
        - » Paramétrage de remplissage correspond à 80% du bloc
  - Il n'est pas possible de compresser toute une table mais certaines colonnes d'une table
    - Cela se fait pour les colonnes contenant des données volumineuses en utilisant TOAST
      - Une ligne ne peut pas être stockée sur plusieurs blocs
      - Ainsi les valeurs importantes sont compressées ou divisées en plusieurs lignes physiques stockées dans une table associée à la table principale
        - » La table principale est appelée HEAP
        - » La table associée est appelée TOAST (The Oversized-Attribute Storage Technique)

Objets et concepts

- Tables et stockage (suite)
  - Lorsque PostgreSQL rencontre une valeur de colonne trop volumineuse pour que la ligne tienne dans un bloc,
    - Il compresse la valeur
    - Si la ligne tient dans le bloc , il stocke cette ligne dans le bloc de la partie HEAP
    - Si la ligne est trop grande il divise la valeur compressée en plusieurs segments de 2K qu'il stocke segment par segment dans la partie TOAST en lui affectant un OID
    - Il enregistre cet OID dans la partie HEAP pour faire le lien
  - Le fichier TOAST est une table composée de 3 colonnes
    - Chunk_id, contient l'OID stocké dans la partie HEAP
    - Chunk_seq, numéro de séquence
    - Chunk_data, la valeur de la données exportée, compressée ou non

Objets et concepts

- Catalog PostgreSQL
  - La table pg_tables contient la lsite des tables et shemas
    - Select schemaname from pg_tables where relname='clients';
      - Affiche le schéma de la table clients
  - La table pg_class contient les détails sur les tables comme le nombre de colonnes, de contraintes, …
    - Select relnatts from pg_class where relname = 'clients';
  - La table pg_type contient les types de données
  - La table pg_roles contient des informations sur les roles
    - Select typname from pg_type where typowner = (select oid from pg_roles where rolname = 'Charly' and typtype = 'd';
      - Liste des tables de Charly

Objets et concepts

- Index
  - Un index est une tables de référence créée pour optimiser le temps de réponse des requêtes SELECT
  - Il existe plusieurs types d'index dans PostgreSQL
    - B-Tree, utilisé par défaut, structure de données en arbre
    - Hash, ce type d'index ne supporte que les comparaisons avec l'opérateur '=' et sont donc destinés à un usage restreint (non journalisés)
    - R-Tree, utilisé pour les types spatiales à 2 dimensions, remplacé en version 8,2 par le type GisT
    - GisT, infrastructure destinée à intégrer de nombreux types d'index
      - utilisé pour les types spatiales
    - GIN, (Generalized Inverted index) infrastructure permettant l'indexation de paires (clé-valeurs)
      - Conçus pour l'indexation de valeurs composées et la recherche d'un élément particulier parmi un ensemble sur une ligne
    - BRIN, apparu en version 9 , (Block Range Index), conçu pour gérer de grosses tables pour lesquelles certaines colonnes ont une corrélation naturelle avec leur emplacement physique dans la table

Objets et concepts

- L'ordre CREATE INDEX
  - CREATE [UNIQUE] INDEX [CONCURRENTLY] [[IF NOT EXISTS] *nom*] ON *nom_table* [
  - USING *méthode*]
  - ({ *nom_colonne* | (*expression*) } [COLLATE *collation*] [*classeop*] [ASC |
  - DESC] [NULLS { FIRST | LAST }] [, ...])
  - [WITH (*parametre_stockage* = *valeur* [, ...])]
  - [TABLESPACE *nom_espacelogique*]
  - [WHERE *prédicat*] ;

Objets et concepts

- Séquence
  - Une séquence est un compteur utilisable pour l'incrément de colonnes
    - Par exemple une colonne correspondant à une clé primaire
    - La séquence sera un objet créé avant la table et utilisé en précisant sont utilisation au moment de la création de la table
    - Exemples :
      - Create table Employe (id_emp interger serial primary key,
        …);
      - Create sequence sep_id_emp ;
      - Create table Employe (id_emp integer default nexval(seq_id_emp) primary key,
        …);
      - Le premier exemple utilise une séquence par l'option « serial » qui crée automatiquement la séquence et l'utilise avec l'option « nextval ».
      - Le deuxième exemple précise l'utilisation de la séquence explicitement.

Objets et concepts

- L'ordre CREATE SEQUENCE
  - CREATE [TEMPORARY | TEMP] SEQUENCE [IF NOT EXISTS] *nom* [INCREMENT [BY]
  - *incrément*]
  - [MINVALUE *valeurmin* | NO MINVALUE]
  - [MAXVALUE *valeurmax* | NO MAXVALUE]
  - [START [WITH] *début*]
  - [CACHE *cache*]
  - [[NO] CYCLE]
  - [OWNED BY { *nom_table*.*nom_colonne* | NONE }];

Objets et concepts

- Procédures stockées
  - Il est possible de stocker dans le serveur PostgreSQL des procédures en utilisant le gestionnaire de procédures
  - PosgreSQL ne gère pas lui-même les langages de procédures et délègue cette fonctionnalité à un gestionnaire dédié
    - Ce qui permet d'implémenter de nombreux langages de procédure stockées
      - Par exemple PL/JAVA, PL/PHP qui s'appuie sur un gestionnaire externe au serveur
    - Le langage PL/pgSQL est livré avec le noyau PostgreSQL

Objets et concepts

- Procédures stockées
  - Les langages doivent être installés dans chaque base de données car il ne le sont pas par défaut
    - Pour cela utiliser la commande CREATE LANGAGE
    - Ou avec la commande système createlang
  - Il est possible d'installer un langage dans une base modèle pour pouvoir ensuite en disposer après création d'une base de données à partir de cette base modèle
  - Il est également possible d'activer le langage PL/pgSQL avec la commande
    - Create langage plpgsql ;
    - OU
    - Createlang plpgsql nombase ;
  - L'avantage du langage PL/pgSQL est d'être proche du langage SQL et d'être compatible avec les types de données, les opérateurs et les fonctions du SQL
    - Une procédure écrite en PL/pgSQL est portable sur tous les moteurs PostgreSQL

Objets et concepts

- Règles
  - Système de réécriture des requêtes appelé règles de réécriture, qui permet de modifier en profondeur l'interprétation des ordres SQL
    - Ainsi une règle permet de détourner une action afin d'en effectuer une autre à la place
  - Exemple
    - Remplacer la suppression d'un article par la mise en place d'une suppression logique en ajoutant une colonne et remplacer l'ordre DELETE par la mise à jour de cette colonne

Travaux pratiques

- Dans le cahier de travaux pratiques
  - Faire l'exercice 03_créer objets
    - Commencez par le script 02_creetables
    - Puis par le script 03_insertion_lignes
  - Pour cela utiliser le répertoire :
    - TP_Postgres\02_creeobjets