SAE : SECURITE DE DONNEES

Annee 2025

- TOURE Ibrahima

UCA

Pr: KEVIN Attigheci

Résume

Le script SQL proposé commence par la création de comptes utilisateurs pour différents profils du personnel hospitalier, avec des mots de passe sécurisés. Des rôles ont ensuite été définis et organisés de manière hiérarchique (par exemple : les médecins héritent des droits des infirmiers).

Des droits précis ont été accordés à chaque rôle sur les tables de la base :

- Les **administrateurs** ont un contrôle total sur la base (création de tables, suppression, etc.).
- Les **médecins** peuvent lire et modifier les diagnostics, gérer les rendez-vous et consulter les données de référence.
- Les **infirmiers** ont des droits limités à la mise à jour des traitements et au suivi des patients.
- Les **patients** ont un accès restreint via une vue sécurisée qui limite les informations visibles.

Enfin, une politique de sécurité de base a été mise en place en révoquant tous les droits par défaut, afin de s'assurer qu'aucun utilisateur n'ait accès à des données sensibles sans autorisation explicite.

Des tests de connexion simulant différentes sessions ont été réalisés pour vérifier que les droits attribués fonctionnent correctement : certaines actions réussissent comme prévu, d'autres échouent logiquement par manque de permission. Cela garantit que la politique de sécurité fonctionne comme attendue.

I. Introduction

Dans le cadre de la gestion d'un système hospitalier sous PostgreSQL, il est essentiel de mettre en place une structure claire pour les utilisateurs et les permissions. Ce projet vise à sécuriser l'accès aux données médicales sensibles tout en respectant les rôles réels dans un hôpital : médecins, infirmiers, patients, et administrateurs. L'objectif est de permettre à chaque utilisateur d'accéder uniquement aux informations dont il a besoin pour effectuer son travail, tout en bloquant l'accès aux données confidentielles ou inutiles.

II. Les problèmes rencontrés lors de la création des tables et de l'insertion des données

Au début, j'ai créé les tables en utilisant directement les scripts fournis par le projet, sans savoir qu'il fallait les adapter pour qu'ils soient compatibles avec PostgreSQL. Cela m'a causé beaucoup d'erreurs, notamment à cause de la syntaxe différente entre les systèmes. Par exemple, PostgreSQL n'accepte pas certains types comme VARCHAR2, ni les guillemets doubles " " pour les chaînes de caractères, et exige un format de date précis comme 'YYYY-MM-DD'. Ensuite, j'ai corrigé les scripts de création pour qu'ils soient réellement bien compris par PostgreSQL. Après cela, j'ai voulu insérer les données en utilisant les scripts d'insertion originaux (ceux fourni par le projet). Mais là aussi, j'ai eu de nombreuses erreurs, notamment à cause du mauvais format des dates, des guillemets, ou encore des noms d'hôpitaux mal écrits ou incohérents avec les tables. J'ai donc dû restructurer les insertions en faisant attention à créer d'abord les tables principales (comme HOSPITAL et RECEPTION) avant les tables dépendantes (comme PATIENT ou APPOINTMENT) pour respecter les contraintes de clé étrangère. Par exemple, il fallait que RECEPTION soit insérée avant APPOINTMENT, sinon PostgreSQL refusait les insertions. Grâce à cette correction étape par étape, mes scripts sont maintenant fonctionnels dans PostgreSQL.

III.Création de compte utilisateurs et leurs rôles

Dans cette partie, j'ai créé trois comptes différents pour trois personnes de l'hôpital : un médecin (dr_dupont), un infirmier (infirmier_leclerc) et un administrateur (admin_hopital). Chaque personne a son propre identifiant et un mot de passe sécurisé.
C'est important parce que cela permet de savoir qui se connecte et qui fait quoi dans la base de données.
Avoir des mots de passe solides aide aussi à protéger les données contre les accès non autorisés.

1. Définir les rôles selon le métier

Ensuite, j'ai défini des rôles qui correspondent aux métiers de l'hôpital : médecin, infirmier, patient, administration. J'ai aussi ajouté des rôles plus spécifiques comme chirurgien et pédiatre. Créer ces rôles permet de regrouper les droits par métier. C'est beaucoup plus facile à gérer que de donner les droits un par un à chaque utilisateur. Par exemple, tous les médecins auront les mêmes permissions, peu importe la personne.

2. Associer les rôles aux utilisateurs

Une fois les rôles créés, je les ai associés à chaque compte.Par exemple, le compte dr_dupont a reçu les rôles medecin et chirurgien. L'infirmier_leclerc a reçu le rôle infirmier.
Et l'administrateur admin_hopital a reçu le rôle administration. Cela permet à chaque personne d'avoir les droits dont elle a besoin selon son métier.On peut aussi donner plusieurs rôles à un même utilisateur si nécessaire.

3. Organiser les rôles de façons hiérarchiques

Pour éviter de répéter les mêmes droits, j'ai relié les rôles entre eux.Par exemple, les médecins ont automatiquement les droits des infirmiers.Le rôle medecin contient aussi le rôle pediatre.Et le rôle administration a tous les droits des médecins.Avec cette organisation, on n'a pas besoin de tout reconfigurer à chaque fois. C'est plus rapide et plus propre à gérer, surtout si on ajoute de nouveaux utilisateurs plus tard.

VI. Attributiuon des privilèges aux différents rôles

L'attribution des privilèges permet de définir ce que chaque rôle a le droit de faire dans la base de données. Par exemple, un médecin peut modifier des diagnostics, alors qu'un patient peut seulement consulter certaines informations personnelles. Ces droits sont adaptés à chaque fonction dans l'hôpital. Cela permet de mieux protéger les données sensibles, de limiter les erreurs, et d'assurer que chaque utilisateur accède uniquement aux informations dont il a réellement besoin pour son travail. Grâce à cette organisation, la base est plus sûre, plus claire et mieux structurée.

1. Privilèges pour l'administrateur

Le rôle administration a tous les droits sur les tables et les séquences de la base de données. Il peut aussi créer des objets comme des tables temporaires. Ce rôle est utile pour gérer l'ensemble du système. L'administrateur doit pouvoir faire des modifications importantes et assurer le bon fonctionnement de la base.

2. Privilèges pour les médecins

Les médecins ont le droit de consulter, ajouter et modifier des informations médicales importantes. Cela concerne les diagnostics, les examens et les rendez-vous. Ils peuvent aussi consulter des informations utiles comme la liste des patients, des médicaments ou des hôpitaux. Par contre, ils ne peuvent pas voir les données financières ou commerciales. Cela permet de bien séparer les rôles médicaux des rôles administratifs ou économiques.

3. Privilèges pour les infirmiers

Les infirmiers peuvent mettre à jour certaines informations comme les traitements ou remarques dans les diagnostics. Ils peuvent aussi consulter les informations des patients, les rendez-vous et les visites. En revanche, ils ne peuvent pas modifier les diagnostics eux-

mêmes, ni changer la date ou le médecin responsable. Cela permet aux infirmiers d'effectuer leur travail tout en gardant certaines décisions médicales réservées aux médecins.

4. Privilèges pour les patients

J'ai créé une **vue spéciale** appelée patient_info pour que les patients puissent voir leurs propres informations. Cette vue affiche uniquement des données de base : nom, prénom, âge, sexe, infirmier référent et identifiant.
Les patients ne peuvent pas voir d'autres informations sensibles.
Cela respecte la confidentialité et les lois comme le RGPD.

V. Sécurité de la base de données

Par défaut, j'ai retiré tous les droits à tous les utilisateurs (REVOKE ALL). Cela s'appelle une politique de sécurité "zéro confiance". Avec ça, personne n'a accès à rien tant qu'on ne lui donne pas les droits expressément. C'est très utile pour éviter les erreurs et les accès non autorisés. Seulles les personnes autorisées reçoivent les permissions nécessaires.

VI. Tester les rôles avec des cas pratiques

Pour vérifier que les droits ont bien été appliqués, j'ai réalisé des tests en me connectant avec chaque compte. Cela permet de s'assurer que chaque utilisateur peut faire uniquement ce qui correspond à son rôle.

D'abord, avec le compte du médecin dr_dupont, j'ai essayé d'accéder à la liste des rendez-vous et de créer un nouveau diagnostic. Ces deux actions ont fonctionné, ce qui montre que le médecin a bien les droits nécessaires pour consulter les informations médicales et en ajouter. En revanche, il n'a pas pu supprimer un diagnostic ni accéder à des données sensibles comme les achats, ce qui est une bonne chose, car ces informations ne sont pas censées être vues par un médecin.

Ensuite, je me suis connecté avec le compte de l'infirmier infirmier_leclerc. J'ai pu modifier un traitement dans un diagnostic et consulter les rendez-vous. Cela montre que l'infirmier a bien accès à certaines informations médicales utiles pour son travail. Cependant, il n'a pas pu changer la date du diagnostic, ce qui prouve que les restrictions ont bien été mises en place pour ne pas lui donner plus de droits que nécessaire.

Enfin, j'ai testé le compte de l'administrateur admin_hopital. Avec ce compte, j'ai pu supprimer une ligne dans la table des visites et créer une nouvelle table pour enregistrer des actions. Cela confirme que l'administrateur a tous les droits sur la base de données et peut donc faire toutes les actions possibles. Ces tests montrent que les droits ont été bien définis pour chaque rôle et que la base de données est bien protégée.

VII. Conlusion

Grâce à ce travail, j'ai appris à créer des utilisateurs, à définir des rôles et à attribuer des droits de manière organisée dans une base de données PostgreSQL. Chaque utilisateur a reçu uniquement les permissions dont il a besoin, ce qui permet de respecter le principe de sécurité et de confidentialité.

La hiérarchie entre les rôles rend la gestion plus simple et plus logique. Par exemple, un médecin peut avoir automatiquement les droits d'un infirmier. Les tests réalisés avec différents comptes montrent que le système fonctionne comme prévu : chacun peut faire ce qu'il doit faire, mais ne peut pas aller au-delà.

Ce type de gestion est très utile, surtout dans un environnement sensible comme un hôpital, où les données doivent être bien protégées et accessibles seulement aux bonnes personnes.

Ce projet m'a permis de mieux comprendre la gestion des accès dans une base de données, un point essentiel en informatique et en science des données.