Institut catholique des arts et métiers

Université catholique d'Afrique centrale - Université Loyola du Congo

Systèmes de gestion de bases de données :

Procédure de construction du prototype SIGGEAC

Modularité — Travail pratique

INFO323_2025-1_S3_TP GROUPE 3 :

DJEGUE TANAWA WILFRIED

TSAJO FILS LOGAN

KANA SALOME LESLY

NOUNGOUA STEVEN

—

— document de travail —

Table des matières

Présentation

Ce travail consiste à mettre en pratique de la notion de modularité appliquée à la conception et à la mise en oeuvre de bases de données dans une perspective de saine gestion du couplage, de la défense contre l'injection, de gestion des accès et la production automatisée de la documentation de conception.

1. Prérequis

- PostgreSQL 17+
- DataGrip 2024.3+ ou autre client SQL
- SchemaSpy 6.2.x
- Java (pour SchemaSpy)
- Un terminal ou éditeur de code ou IDE

2. Structure du prototype

La base SIGGEAC est subdivisée en trois modules (schémas) :

Schéma	**Description**
etudiant	Données personnelles et évaluations
professeur	Données professorales et affectations
offre_service	Données des cours, modules, structures académiques

3. Étapes de construction

3.1. Initialisation de l'architecture de base

Exécuter le contenu du fichier **000_SIGGEAC_ini.sql**

Le script 000_SIGGEAC_ini.sql prépare l'environnement modulaire de la base de données :

- Création des schémas logiques (etud, prof, offre, etc.)
- Définition des premiers rôles applicatifs et techniques

3.2. Création des types ou domaines personnalisés

Exécuter le contenu du fichier **010_SIGGEAC_MDD_types.sql**

Le fichier 010_SIGGEAC_MDD_types.sql définit les types de domaine stricts, améliorant la robustesse et la clarté du modèle conceptuel de données.

Types créés :

- Identifiants : uuid_pk, matricule
- Chaînes typées : nom, prenom, sigle_cours, sigle_module
- Numériques : credit, note_sur_100
- Catégories : cycle, session

3.3. Création du modèle de données

Exécuter le contenu du fichier **020_SIGGEAC_MDD_tables.sql**

Le script 020_SIGGEAC_MDD_tables.sql implémente la structure physique des données, en s'appuyant sur les types personnalisés et les schémas définis.

- Tables étudiants, professeurs, offres de cours, modules, etc.
- Contraintes d'intégrité référentielle
- Séparation fonctionnelle par schéma

3.4 Mise en place des rôles et de la politique d'accès

Exécuter le contenu du fichier **030_SIGGEAC_IMM_base.sql**

Le fichier **IMM_base.sql** met en place une politique d'accès stricte, basée sur des rôles applicatifs et des rôles usagers par domaine fonctionnel.

- Rôles applicatifs : consultation, modification, admin
- Rôles usagers : etudiant_dossier, professeur_affectation, etc.
- Utilisateurs : imm_etudiant, imm_professeur, imm_administrateur
- Droits fins sur chaque schéma et table

3.5 Jeu de données pour tests

Le fichier **data_test.sql** contient des données représentatives permettant de.

- Vérifier l'intégrité du modèle
- Tester les permissions

Simuler les opérations de lecture/écriture réelles

4. Applications minimales de test

Le répertoire nommé **IMM_project** est le projet permettant d'exécuter les applications illustrant chacune des trois interfaces (une relative aux données étudiantes, l'autre aux données professorales, et la troisième à l'offre de service et aux données structurelles) et permettant de les tester

- L'IDE Intellij nous permet de charger le répertoire et exécuter le projet pour lancer les applications (d'autres IDE peuvent être utilises). Exécuter au préalable la commande mvn generate-sources (pour build le projet afin de le lancer ensuite)
- Chaque application cible un schéma précis. Et il n'y a plus qu'à suivre les indications pour effectuer les tests
- Exemple (consultation des étudiants) :

```
#sql

SET ROLE consultation ;

SELECT * FROM etudiant.etudiant WHERE id = 1;
```

5. Déploiement de la documentation SchemaSpy

Le répertoire nommé **output** contient la documentation générée automatiquement à partir de **SchemaSpy**

5.1. Configuration du fichier schemaspy.properties :

- t pgsql \
- dp postgresql-jdbc.jar \
- db postgres \
- host localhost \
- port 5432 \
- u postgres \
- p inconnu_X2027 \
- ./output \
- s public \
- renderer:cairo:scale=2 \
- hq \
- norows

Exécution de SchemaSpy

Exécuter la commande bash : **java -jar schemaspy.jar \configFile**

schemaspy.properties

6. Défense contre l'injection

- Utilisation de vues restreignant les accès.
- Pas de SQL dynamique.
- Scripts d'accès toujours paramétrés (via jOOQ).

7. Migration : différenciation terminale

- Les scripts de mise à jour seront conçus pour ne modifier que ce qui change, en suivant une convention de fichiers migration_YYYYMMDD.sql.

Produit le 2025-05-10

Institut catholique des arts et métiers Université catholique d'Afrique centrale - Université Loyola du Congo