UQAC

Département d'informatique et de mathématiques
8TRD157 – Bases de données avancées

Travail pratique #2 : Conception physique

Étudiant 1	Hajar Chakir
Étudiant 2	
Date	03/11/2025

Rapport	
Contraintes	**/ 35**
Procédures	**/ 35**
Gestion des accès	**/ 10**
Création de tables	**/ 10**
Question théorique	**/ 10**
Total	**/ 100**

1 Règles d'affaires (contraintes)

[Décrivez brièvement chacune des contraintes identifiées pour les cas d'utilisation et précisez la stratégie employée pour implémenter cette contrainte (CHECK, FOREIGN KEY ou TRIGGER).]

Plusieurs règles d'affaires ont été identifiées dans le système et ont été implémentées à l'aide de trois mécanismes : les contraintes CHECK, les clés étrangères et les triggers. Leur objectif est d'assurer la cohérence des données tout en respectant les exigences fonctionnelles du domaine universitaire.

1 Contraintes de domaine (CHECK)

Plusieurs validations simples ont été imposées directement au niveau des colonnes :

- **Statut étudiant** : la valeur doit obligatoirement appartenir à l'ensemble {C, A, T, S, P}.
- **Statut enseignant** : limité aux valeurs numériques 0 à 4.
- **Crédits d'un cours** : le nombre de crédits doit être strictement supérieur à 0.
- **Format du sigle d'un cours** : le sigle doit respecter la structure *[numéro de département][trois lettres][trois chiffres]* (ex. 8TRD157).
- **Format du code local** : imposé via une expression régulière du type *A1-2345* (une lettre, un chiffre, un tiret, puis un chiffre entre 1 et 4 suivi de trois chiffres).
- **Format de la session** : doit respecter le modèle *AAAAT* (par exemple 20241).
- **Format du groupe** : exactement deux chiffres, à partir de « 01 ».
- **Validité des périodes de cours** : le jour de la semaine doit faire partie d'un ensemble limité, et l'heure de fin doit être postérieure à l'heure de début.
- **Prérequis** : un cours ne peut pas être prérequis de lui-même.

Ces contraintes CHECK assurent la qualité et la validité des données sans logique complexe.

2 Contraintes d'intégrité référentielle (FOREIGN KEY)

Les règles structurelles du modèle sont assurées par des clés étrangères, qui garantissent la cohérence des liens entre les entités :

- Une **personne** peut être soit un étudiant, soit un enseignant, via une relation 1-à-1 (NAS partagé).
- Chaque **étudiant** et chaque **enseignant** doit appartenir à un **département** existant.
- Une **adresse** est liée à une **personne**.
- Un **cours** appartient à un **département**, et peut avoir un **enseignant responsable**.

- Une **offre de cours** (COURS_ENSEIGNE) relie un cours à un enseignant pour une session donnée.
- Une **période** associe une offre de cours à un local, en imposant que les deux existent.
- Une **inscription** relie un étudiant à une offre de cours existante.
- Un **cours prérequis** doit référer à un autre cours existant.

Ces contraintes empêchent notamment de créer un étudiant sans département, un local inexistant pour une période, ou encore une inscription sur un cours qui n'a pas été ouvert.

3. Contraintes dynamiques complexes (TRIGGERS)

Certaines règles ne peuvent pas être exprimées par de simples CHECK ou FK. Elles ont été implémentées à l'aide de triggers :

- **Directeur obligatoire** : dès qu'un cours est offert dans un département, celui-ci doit obligatoirement avoir un directeur assigné.
- **Directeur et département cohérents** : le directeur d'un département doit être un enseignant du même département.
- **Responsable de cours cohérent** : un enseignant responsable d'un cours doit appartenir au même département que le cours, et le premier chiffre du sigle doit correspondre au numéro du département.
- **Limite de responsabilité** : un enseignant ne peut pas être responsable de plus de six cours.
- **Génération automatique du courriel institutionnel** :
  - pour les enseignants : format *prenom_nom[n]@uqac.ca*,
  - pour les étudiants : format *prenom.nom[n]@uqac.ca*,
    avec incrément automatique en cas de doublon.

Ces triggers encapsulent la logique métier qui ne peut pas être imposée uniquement par la structure des tables.

Pour **résumer**, les contraintes mises en place, qu'elles soient statiques (CHECK), relationnelles (FOREIGN KEY) ou dynamiques (TRIGGER), permettent d'assurer la validité, la cohérence et le respect des règles de gestion définies dans la mise en situation. Elles garantissent la qualité du système tout en facilitant l'encapsulation et la maintenance.

2 Opérations à encapsuler

[Décrivez brièvement les principales opérations identifiées pour les cas d'utilisation]

Les cas d'utilisation du système font intervenir plusieurs opérations de gestion qui modifient l'état de la base de données. Afin d'assurer la cohérence des données et d'éviter les duplications de logique, ces opérations ont été encapsulées dans des procédures stockées.

Les principales opérations sont les suivantes :

- **Création des entités principales** : ajout d'une personne, d'un enseignant, d'un étudiant et d'un département. Chaque procédure centralise l'insertion des données et garantit le respect des contraintes associées (ex. appartenance à un département).
- **Gestion des cours** : création d'un cours et définition éventuelle de son enseignant responsable. L'encapsulation permet de valider automatiquement les règles (ex. cohérence avec le département).
- **Gestion de la structure académique** : affectation d'un directeur de département et ajout de prérequis entre cours. Ces opérations nécessitent des vérifications qui sont regroupées dans des procédures dédiées.
- **Gestion des offres de cours** : ouverture d'un cours pour une session donnée (cours enseigné), incluant session, groupe et enseignant assigné.
- **Inscription d'un étudiant** : ajout d'une inscription en utilisant des paramètres « métier » plutôt que des identifiants internes, afin de simplifier l'usage de la procédure.

Pour **résumer**, les opérations encapsulées couvrent l'ensemble du cycle de gestion des personnes, des départements, des enseignants, des étudiants, des cours et des inscriptions. Ces procédures permettent d'encapsuler toute la logique d'insertion et de mise à jour, d'éviter les erreurs manuelles et d'assurer automatiquement le respect des règles d'affaires du système.

3 Planification des tâches

[Décrivez brièvement comment le travail a été divisé dans votre équipe. Estimez, pour chaque tâche de l'énoncé, le pourcentage du travail effectué par chacun des membres de votre équipe.]

Je n'ai pas de binôme et j'ai travaillé seul.

4 Question théorique

[Créez une vue (VIEW) permettant d'illustrer le concept d'encapsulation (masquer la complexité d'une opération ou requête) dans votre schéma. Cette vue doit permettre d'afficher l'ensemble de l'information stockée relative à un département (information du département ET de l'enseignant qui est directeur (son nom et son prénom). Donnez le code SQL permettant de créer cette vue.]

L'encapsulation consiste à masquer la complexité d'une opération ou d'une requête tout en offrant à l'utilisateur une interface simple et cohérente. Dans le contexte du schéma de données, une vue (VIEW) permet d'encapsuler plusieurs jointures en une seule structure logique facile à interroger.

La vue suivante illustre ce principe en regroupant, dans un même objet, l'ensemble des informations d'un département ainsi que les informations de l'enseignant qui en est le directeur (nom et prénom). L'utilisateur peut consulter cette vue sans connaître la structure des tables ni les relations entre celles-ci.

```
CREATE OR REPLACE VIEW V_INFO_DEPARTEMENT AS
SELECT
    d.noDepartement,
    d.nomDepartement,
    d.NAS_Directeur,
    p.nom    AS nomDirecteur,
    p.prenom AS prenomDirecteur
FROM DEPARTEMENT d
LEFT JOIN ENSEIGNANT e
       ON d.NAS_Directeur = e.NAS
LEFT JOIN PERSONNE p
       ON e.NAS = p.NAS;
```

Cette vue encapsule ainsi la complexité des jointures entre les tables DEPARTEMENT, ENSEIGNANT et PERSONNE, et permet d'accéder directement à l'information complète d'un département à l'aide d'une seule requête :

SELECT * FROM V_INFO_DEPARTEMENT;

5 Gestion des droits d'accès

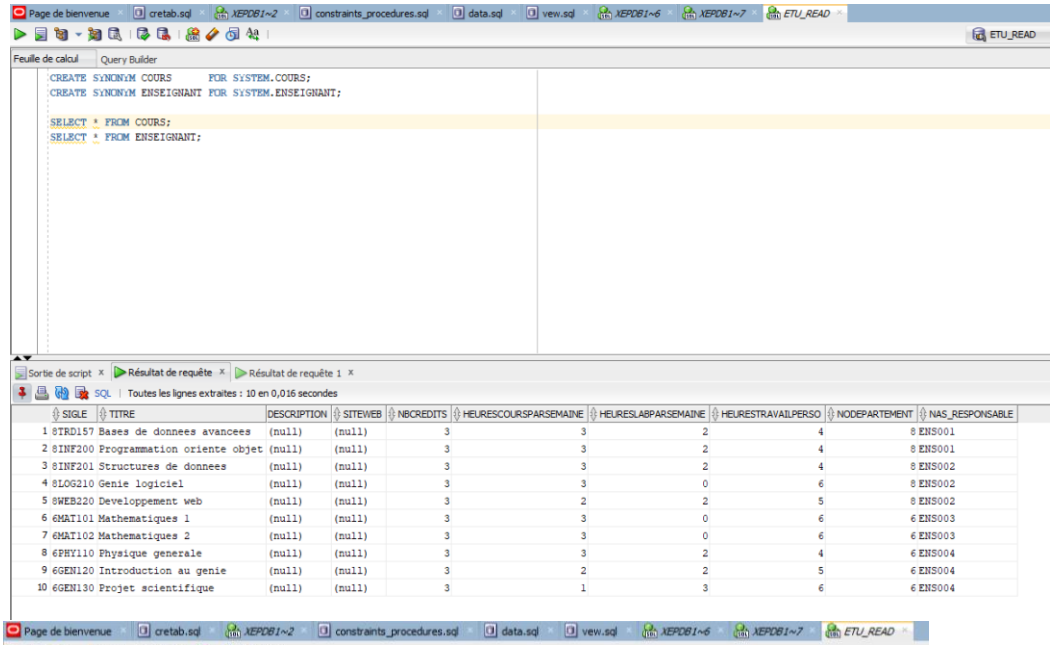
[Veuillez inclure les 3 captures d'écrans demandées dans la tâche 4 de l'énoncé.]

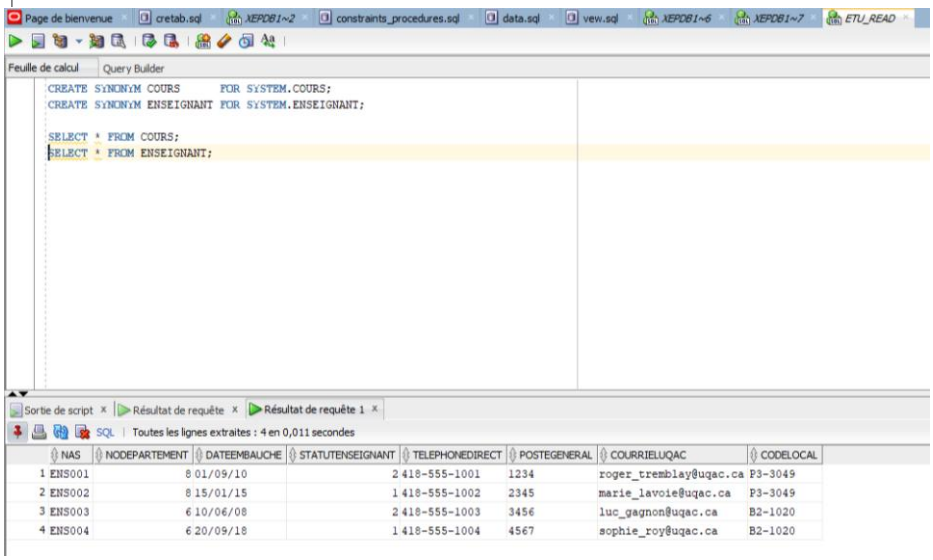
Dans cette tâche, deux utilisateurs distincts ont été configurés dans Oracle :

- **Utilisateur administrateur** : utilisé pour créer les tables, contraintes, triggers et pour insérer les données (Tâches 1 à 3).
- **Utilisateur standard (ETU_READ)** : possède uniquement le privilège SELECT sur les tables du schéma. Cet utilisateur ne peut ni insérer, ni modifier, ni supprimer des données.

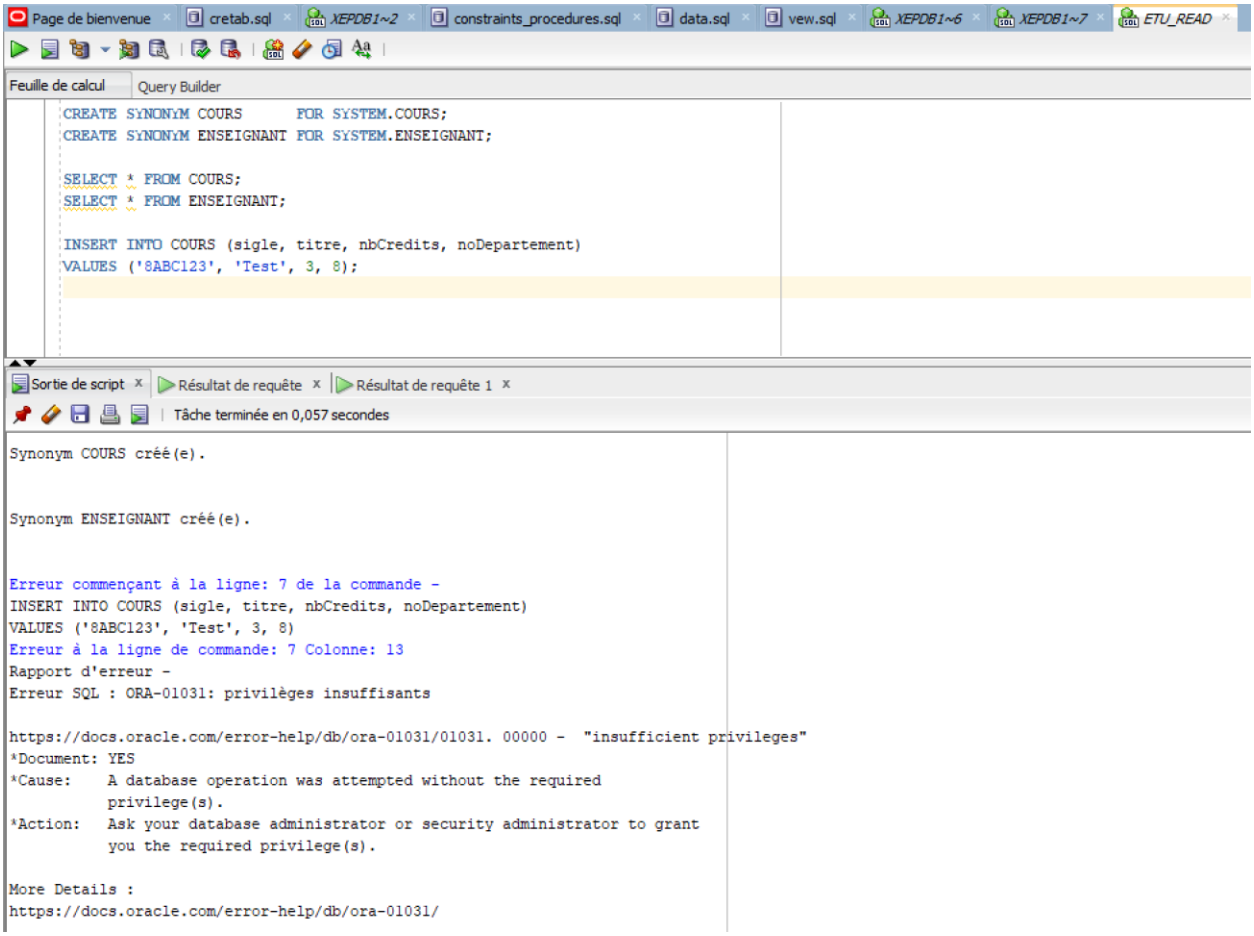
Les captures d'écran ci-dessous illustrent le comportement attendu :

1. **Consultation des données avec l'utilisateur standard** : L'utilisateur ETU_READ peut exécuter la requête SELECT * FROM COURS et accéder aux données normalement.

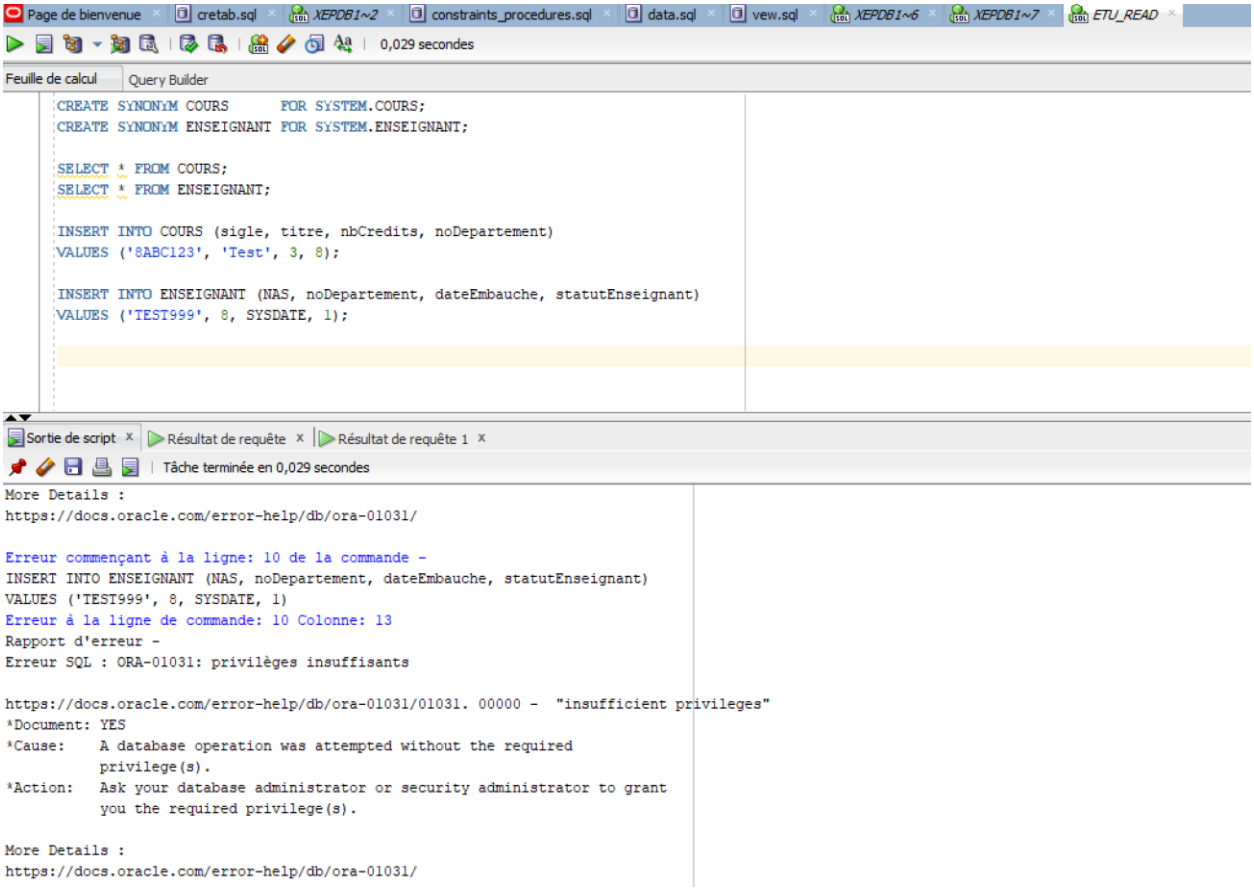




2. **Tentative d'insertion d'un cours** : Lorsqu'un INSERT INTO COURS est exécuté avec l'utilisateur standard, Oracle retourne l'erreur ORA-01031: insufficient privileges, confirmant l'absence de droits d'écriture.



3. **Tentative d'insertion d'un enseignant** : La même erreur apparaît lors d'un INSERT INTO ENSEIGNANT, ce qui confirme que l'utilisateur standard n'a aucun privilège de modification.



Ces résultats démontrent que la gestion des droits d'accès a été correctement mise en place et que l'utilisateur standard dispose uniquement de privilèges en lecture, comme exigé dans les consignes.