FORMATION DEV WEB WEB MOBILE
CCP 2 - DÉVELOPPER LA PARTIE BACK-END D'UNE APPLICATION WEB

LES BASES DE DONNEES RELATIONNELLES

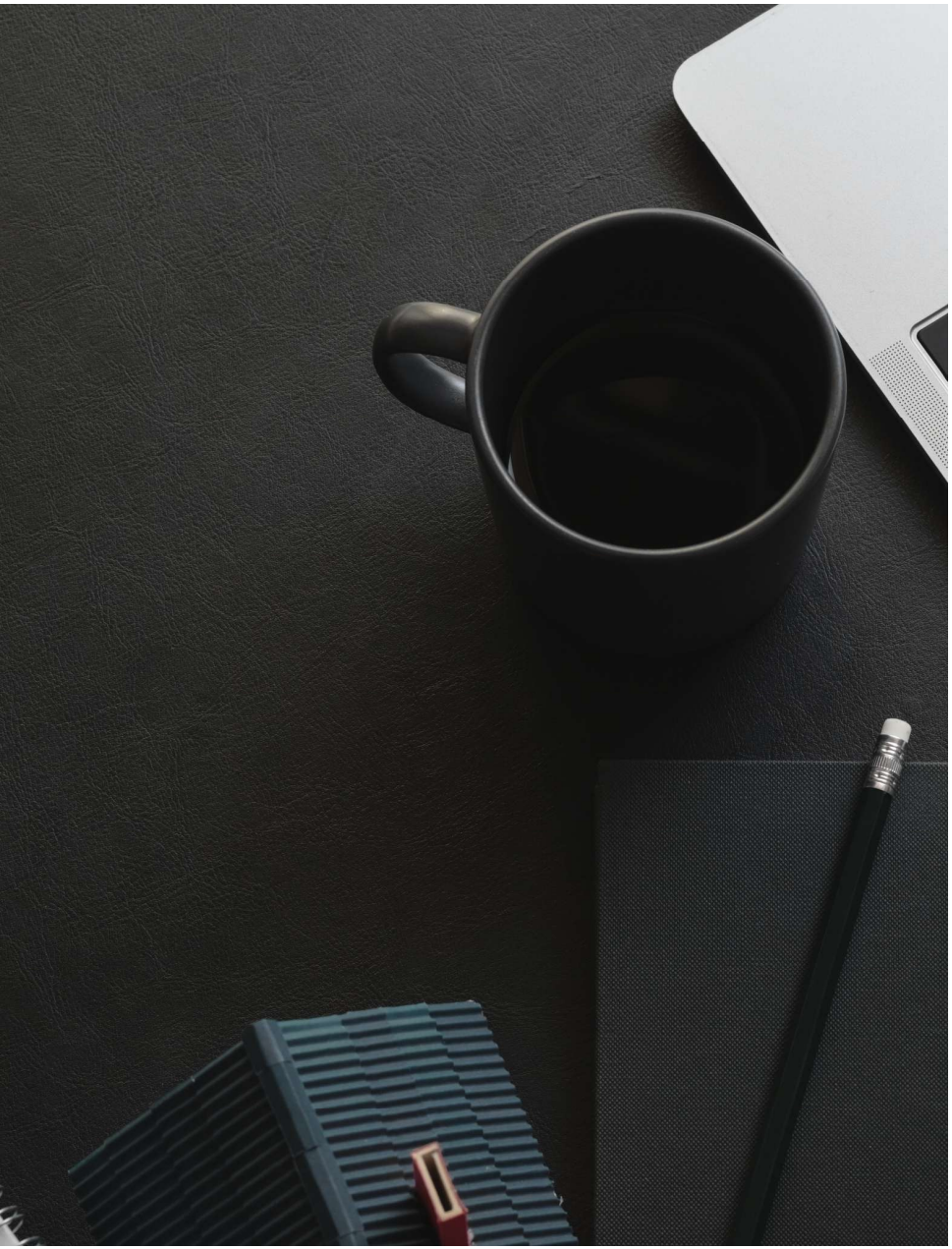
JOUR 3 : CONTRAINTES ET SCHEMA PHYSIQUE

PRESENTE PAR NATACHA DESSE

AFEC

DEROULE MODULE

- Découverte des BDD et Installation
- Les relations et modélisation (DEA)
- **Contraintes et schéma physique**
- Jeu d’essai, insertion et agrégations, requêtes avancées
- Jointures et sous requêtes
- Sauvegardes SQL et gestion des rôles



CONTRAINTES ET SCHEMA PHYSIQUE

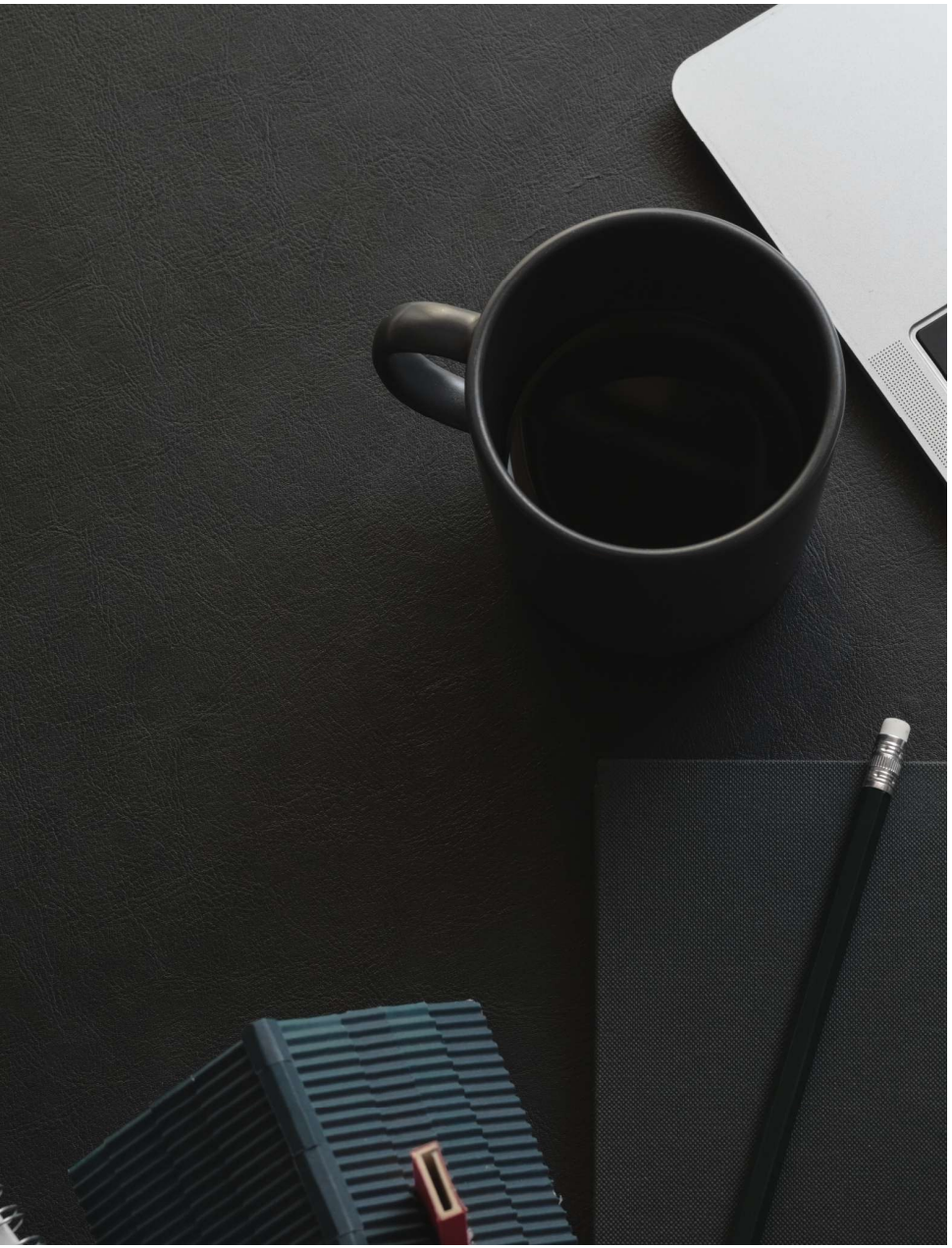
LES OBJECTIFS

Comprendre et appliquer les **contraintes** d'intégrité en SQL

Passer d'un **modèle logique** à un **schéma physique** cohérent

Optimiser la base avec des **index**

Tester et améliorer les **performances**



LES NOTIONS

NUAGE DE MOTS

https://www.menti.com/aldpp4773piz

RAPPEL DU SCHEMA LOGIQUE AU SCHEMA PHYSIQUE

01

Schéma logique

- Représente les entités, relations, attributs, cardinalités
- Ne dépend pas d'un SGBD (niveau conceptuel)

02

Schéma physique

- Implémentation concrète dans un SGBD SQL
- Précise :
  - Types de données
  - Contraintes
  - Index
  - Stockage / partitions éventuelles

LES CONTRAINTES D'INTEGRITE

Les contraintes garantissent la **cohérence** et l'**exactitude** des données

Pourquoi sont-elles importantes ?

- **Éviter** les **erreurs** de saisie (ex : champs vides, doublons)
- Préserver les **liens logiques** entre les tables (ex : client → commande)
- Assurer la **validité** des données (ex : un âge ne peut pas être négatif)
- Renforcer la **qualité** des bases de données sur le long terme
- Rendre les données **compréhensibles** et **utilisables** par tous

NOT NULL

Empêche qu'un champ soit vide

```
nom VARCHAR(100) NOT NULL
```

UNIQUE

Empêche les doublons (sauf NULL)

```
email VARCHAR(100) UNIQUE
```

CHECK

Définit une règle métier sur un champ

```
note DECIMAL(3,1) CHECK (note BETWEEN 0 AND 10)
```

DEFAULT

Donne une valeur par défaut si rien n’est précisé

```
status VARCHAR(10) DEFAULT ‘Actif’
```

PRIMARY KEY

Identifie de façon unique une ligne
Combine NOT NULL + UNIQUE

```
id INT PRIMARY KEY
```

FOREIGN KEY

Assure la cohérence entre deux tables

```
FOREIGN KEY (id_client) REFERENCES Client(id)
```

ATELIER

Ajout de contraintes

CONSIGNES

À partir de cette table brute, ajouter les bonnes contraintes :

```
CREATE TABLE Film (
 id INT,
 titre TEXT,
 note DECIMAL,
 statut TEXT
);
```

RAPPEL

Nom	Rôle
PRIMARY KEY	Identifie de façon **unique** chaque ligne dans une table
FOREIGN KEY	Assure la **cohérence** entre deux tables liées
UNIQUE	Empêche les **doublons** dans une colonne
NOT NULL	Interdit les **valeurs vides**
CHECK	Imposent une **condition logique** sur les valeurs
DEFAULT	Définit une **valeur par défaut** si aucune valeur n'est fournie

CORRECTION

```sql
CREATE TABLE Film (
  id INT PRIMARY KEY,
  titre TEXT NOT NULL,
  note DECIMAL(3,1) CHECK (note BETWEEN 0 AND 10),
  statut TEXT DEFAULT 'À l'affiche'
);
```

MODIFIER UNE CONTRAINTE NOT NULL

Ajouter une contrainte NOT NULL

```
ALTER TABLE nom_table
MODIFY nom_colonne type_de_donnée NOT NULL;
```

Supprimer une contrainte NOT NULL

```
ALTER TABLE Utilisateur
MODIFY email VARCHAR(255) NULL;
```

AJOUTER/SUPPRIMER UNE CONTRAINTE UNIQUE

Ajouter une contrainte UNIQUE

```
ALTER TABLE Utilisateur
ADD CONSTRAINT unique_email UNIQUE (email);
```

Supprimer une contrainte NOT NULL

```
ALTER TABLE Utilisateur
DROP CONSTRAINT unique_email;
```

MODIFIER UNE CONTRAINTE

Le SGBD (comme MySQL, PostgreSQL, etc.) va générer un nom automatique, par exemple users_email_key ou users_email_unique_1.

Mais si tu veux la supprimer plus tard, tu devras :

- d'abord retrouver son nom avec une commande comme :

```
SHOW CREATE TABLE nom_table;
```

```
Table | Create Table
------+-------------------------------------------------------
users | CREATE TABLE `users` (
          `id` int NOT NULL AUTO_INCREMENT,
          `age` int DEFAULT NULL,
          `email` varchar(255) DEFAULT NULL,
          `name` varchar(100) DEFAULT NULL,
          PRIMARY KEY (`id`),
          UNIQUE KEY `email` (`email`)
          CONSTRAINT `users_chk_1` CHECK ((`age` >= 18))
        ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
);
```

Puis utiliser ce nom dans

```
ALTER TABLE nom_table DROP CONSTRAINT nom_contrainte;
```

AJOUTER / SUPPRIMER FOREIGN KEY

Ajouter une clé étrangère

```
ALTER TABLE Commande
ADD CONSTRAINT fk_utilisateur_id
FOREIGN KEY (utilisateur_id) REFERENCES
Utilisateur(id);
```

Retirer la clé étrangère avec le nom de la contrainte

```
ALTER TABLE Commande
DROP CONSTRAINT fk_utilisateur_id;
```

AJOUTER / SUPPRIMER CHECK

Ajouter un CHECK

```
ALTER TABLE Utilisateur
ADD CONSTRAINT verif_age CHECK (age >= 18);
```

Retirer le CHECK avec le nom de la contrainte

```
ALTER TABLE Utilisateur
DROP CONSTRAINT verif_age;
```

ATELIER

Modifications de contraintes

CONSIGNES

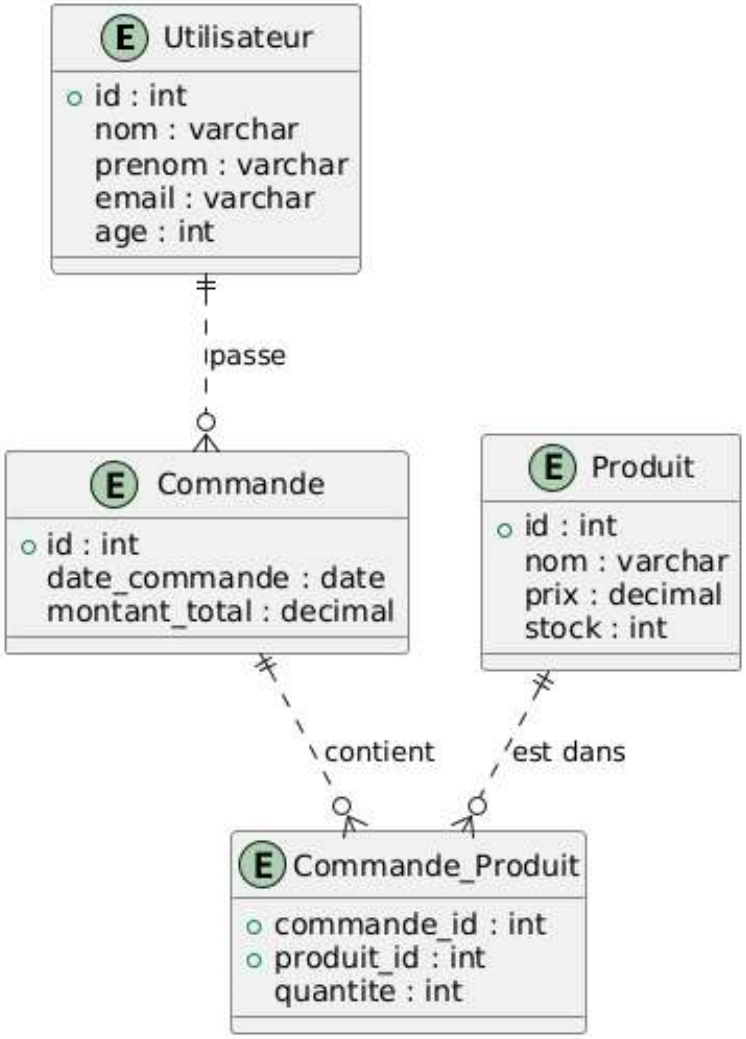
https://github.com/dessna12/03-Constraints-SQL

Cette base contient des erreurs ou oublis de contraintes.

À vous de :

- *repérer ce qui manque ou est incorrect ;*
- *proposer les bonnes contraintes à ajouter ou modifier ;*
- *exécuter les commandes nécessaires pour corriger les problèmes.*

AIDE


Utilisateur
id : int
nom : varchar
prenom : varchar
email : varchar
age : int
passe
Commande
id : int
date_commande : date
montant_total : decimal
Produit
id : int
nom : varchar
prix : decimal
stock : int
contient
est dans
Commande_Produit
commande_id : int
produit_id : int
quantite : int

ERREURS A CORRIGER

- Ajouter NOT NULL UNIQUE(email) à utilisateurs.
- Ajouter NOT NULL sur age et un CHECK(age >= 18).
- Ajouter CHECK(stock >= 0) dans produits.
- Corriger la clé étrangère erronée dans commandes (REFERENCES produits(id) → doit être REFERENCES utilisateurs(id)).
- Ajouter la clé primaire composite dans commande_produit(commande_id, produit_id).
- Ajouter les bonnes foreign keys dans commande_produit.

TYPES ET INDEXATION

LES TYPES CHOIX STRATEGIQUES

Pourquoi bien choisir ?

- Réduction de la taille du stockage
- Accélération des requêtes
- Amélioration de la clarté métier

Type SQL	**Usage courant**
INT	Nombres entiers (souvent pour identifiants)
BIGINT	Entiers très grands (ex. identifiants globaux, compteurs)
SMALLINT	Entiers petits (optimisation mémoire)
VARCHAR(n)	Chaînes de texte de longueur variable
CHAR(n)	Chaînes de texte de longueur fixe
TEXT	Texte long (ex. descriptions, commentaires)
DATE	Dates (année, mois, jour)
TIME	Heures (heure, minute, seconde)
DATETIME	Combinaison date + heure (timestamp)
DECIMAL(x,y)	Nombres décimaux précis (valeurs monétaires, notes)
FLOAT / REAL	Nombres à virgule flottante (moins précis que DECIMAL)
BOOLEAN	Valeurs logiques (vrai / faux)
ENUM	Liste de valeurs définies (ex : statut = 'actif', 'inactif')
BLOB	Données binaires (images, fichiers, etc.)

INDEXATION

Un index est une structure de données permettant d'accélérer l'accès aux lignes d'une table selon certaines colonnes.

- Amélioration des performances des requêtes SELECT.
- Accès rapide à des lignes spécifiques.
- Utilisation efficace dans les clauses WHERE, JOIN, ORDER BY.

⚠️ Attention :

- Trop d'index peut ralentir les INSERT, UPDATE, DELETE.
- Chaque index prend de l'espace mémoire.

TYPE D'INDEX

Type	Description
Index simple	Sur une seule colonne.
Index composite	Sur plusieurs colonnes.
Index unique	Empêche les doublons.
Index plein texte	Optimisé pour les recherches dans du texte long.

SOUS LE CAPOT

Les données d'un index sont triées. Si un index est multicolonne, le tri de la clef d'index fait que l'information y est vectorisée.

Chaque colonne supplémentaire précise la colonne précédente. De ce fait, la recherche dans un index multicolonne n'est accélérée que si les données cherchées sont un sous-ensemble ordonné du vecteur.

Dans cet index composé d'un nom d'un prénom et d'une date de naissance, la recherche sera efficace pour les informations suivantes :

CLI_NOM
CLI_NOM + CLI_PRENOM
CLI_NOM + CLI_PRENOM + CLI_DATE_NAISSANCE

CLI_NOM	CLI_PRENOM	CLI_DATE_NAISSANCE
DUPONT	Alain	21/01/1930
DUPONT	Marcel	21/01/1930
DUPONT	Marcel	01/06/1971
DUPONT	Paul	21/01/1930
MARTIN	Alain	11/02/1987
MARTIN	Marcel	21/01/1930

En revanche, la recherche sur une seule colonne CLI_PRENOM ou CLI_DATE_NAISANCE et a fortiori sur ces deux colonnes n'aura aucune efficacité du fait du tri relatif des colonnes entre elles (vectorisation)

CAS D’UTILISATION

Indexer les colonnes souvent utilisées dans :

- WHERE
- JOIN
- ORDER BY

Éviter d’indexer :

- Colonnes rarement filtrées.
- Colonnes très modifiées.

EXEMPLE D'INDEX SIMPLE

```
CREATE INDEX idx_nom ON Utilisateur(nom);
```

Effet : Accélère les recherches :

```
SELECT * FROM Utilisateur WHERE nom = 'Martin'
```

EXEMPLE D'INDEX COMPOSITE

```
CREATE INDEX idx_nom ON Utilisateur(nom, prenom);
```

Utilisation optimale si la requête filtre sur

- nom
- nom et prénom

Attention pas optimal sur **prénom seul**

SUPRESSION DES INDEX

```
DROP INDEX idx_nom ON Utilisateur;
```

Pour vérifier les index existants

```
SHOW INDEX FROM Utilisateur;
```

ATELIER

Ajouter les bons index

CONSIGNE

Ajouter un index sur la colonne id_client de Commande

```
CREATE TABLE Client (
 id INT PRIMARY KEY,
 nom TEXT
);

CREATE TABLE Commande (
 id INT PRIMARY KEY,
 id_client INT,
 date_commande DATE,
 FOREIGN KEY (id_client) REFERENCES Client(id)
);
```

SOLUTION

```
CREATE INDEX idx_commande_client ON Commande(id_client);
```

ATELIER

Ajout des contraintes et index sur “cineclub”

QUIZZ

QUESTION 1

Que permet la contrainte NOT NULL ?

A) D'interdire les doublons dans une colonne
B) De rendre un champ obligatoire
C) De lier deux tables entre elles
D) De définir une valeur par défaut

QUESTION 2

2. Laquelle des contraintes suivantes garantit l'unicité des valeurs mais autorise les valeurs NULL ?

A) PRIMARY KEY
B) NOT NULL
C) UNIQUE
D) FOREIGN KEY

QUESTION 3

3. Quelle contrainte permet de forcer une règle métier comme "la note doit être comprise entre 0 et 10" ?

A) CHECK
B) DEFAULT
C) UNIQUE
D) NOT NULL

QUESTION 4

La contrainte DEFAULT s’active quand :

A) Une valeur interdite est insérée
B) Aucune valeur n’est précisée pour le champ
C) Une valeur NULL est insérée
D) On veut empêcher les doublons

QUESTION 5

5. Quelle est la différence entre PRIMARY KEY et UNIQUE ?

A) UNIQUE interdit les doublons, PRIMARY KEY non
B) PRIMARY KEY autorise les NULL, UNIQUE non
C) PRIMARY KEY combine UNIQUE et NOT NULL
D) Il n'y a aucune différence

QUESTION 6

6. Quel est le rôle d’une FOREIGN KEY ?

A) Améliorer la performance
B) Appliquer une règle métier
C) Relier une table à une autre
D) Empêcher les données dupliquées

QUESTION 7

7. Sur quel champ est-il pertinent de créer un index ?

A) Un champ rarement utilisé
B) Un champ utilisé dans des WHERE fréquents
C) Un champ textuel très long
D) Un champ de clé primaire uniquement

QUESTION 8

8. Quel est l'impact négatif d'un trop grand nombre d'index ?

A) Ralentissement des lectures
B) Données corrompues
C) Augmentation de la taille et lenteur en insertion
D) Empêche les jointures

QUESTION 9

9. Quelle commande permet d'analyser les performances d'une requête SQL ?

A) CHECK
B) EXPLAIN
C) SELECT *
D) ANALYZE TABLE

QUESTION 10

10. Un index est créé automatiquement sur...

A) Chaque colonne d'une table
B) Les colonnes avec UNIQUE
C) Les colonnes FOREIGN KEY
D) Les PRIMARY KEY

CORRECTION

1. B
2. C
3. A
4. B
5. C
6. C
7. B
8. C
9. B
10. D