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

LES BASES DE DONNEES RELATIONNELLES

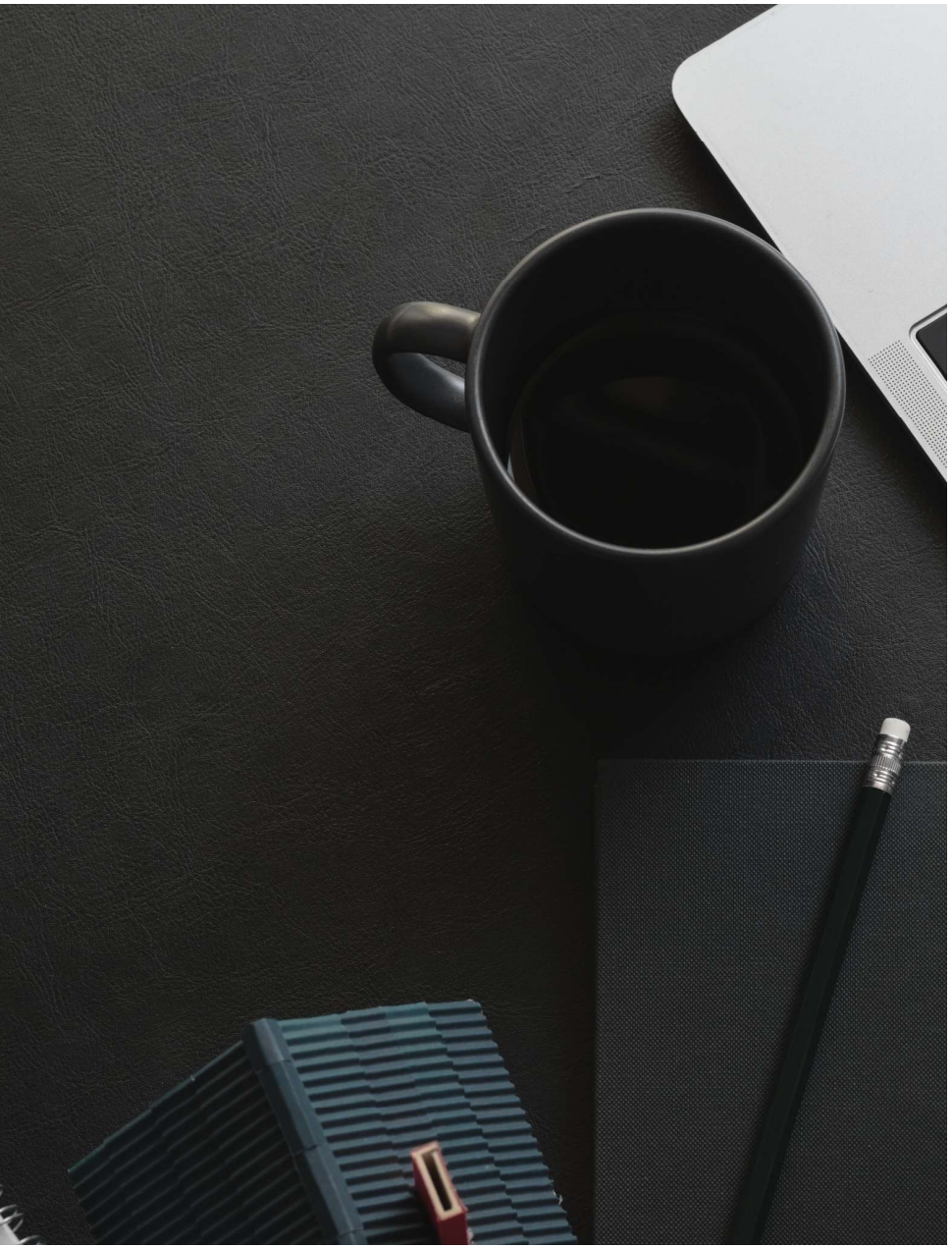
JOUR 6 : SAUVEGARDES SQL ET GESTION DES ROLES

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**



SAUVEGARDES SQL ET GESTION DES ROLES

LES OBJECTIFS

Réaliser des sauvegardes et restaurations de base de données MySQL

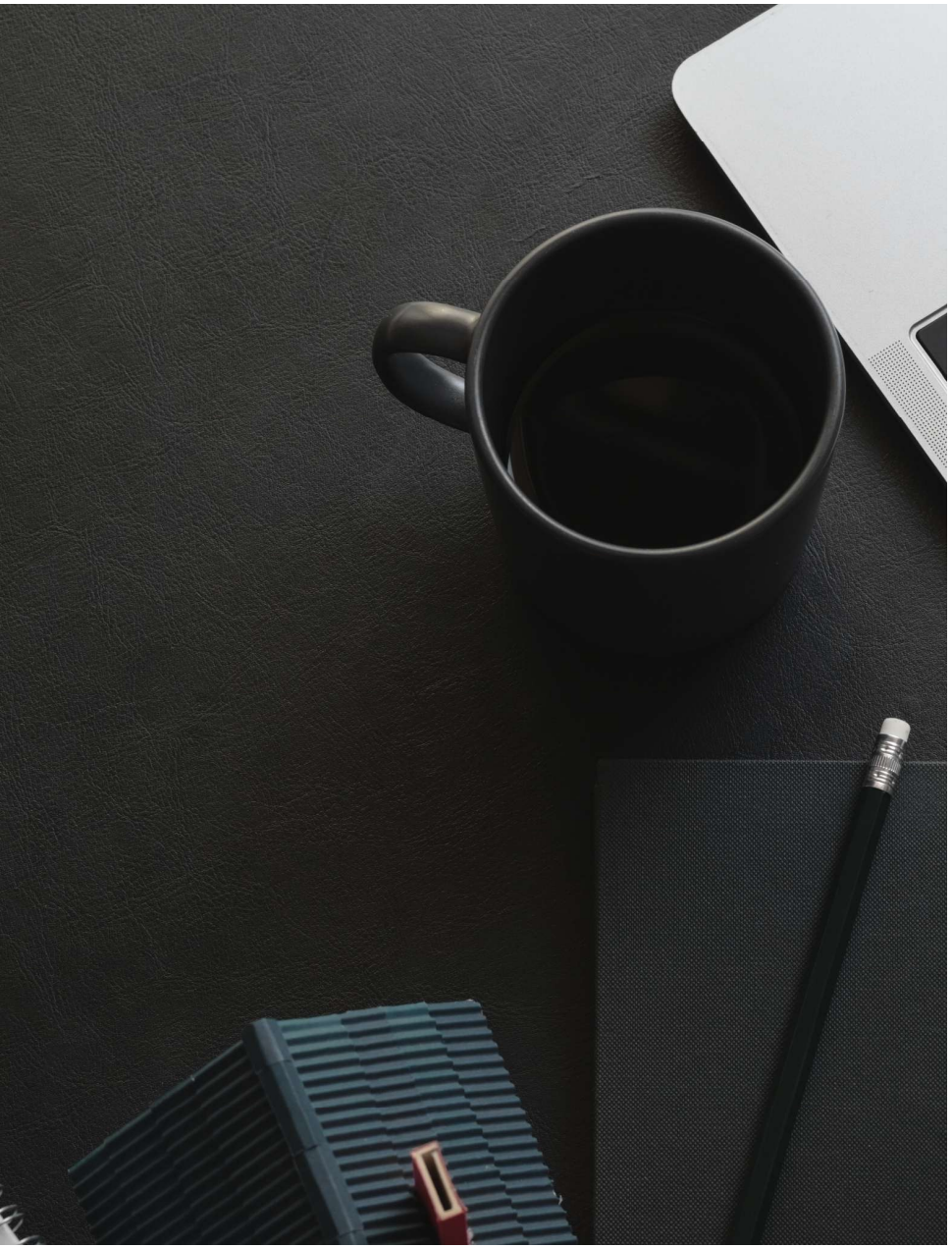
Comprendre la différence entre les types de sauvegardes

Gérer les utilisateurs et leurs permissions

Appliquer les bonnes pratiques de sécurité via les rôles

NUAGE DE MOTS

Qu’avez vous retenu en SQL depuis le début ?



SAUVEGARDES SQL

SCENARIO : UNE JOURNEE ORDINAIRE … JUSQU'AU DRAME

Nous sommes lundi matin, 9h.

L'entreprise VeloMax, une PME spécialisée dans la location de vélos, démarre sa semaine.

Elle utilise une application web maison pour :

- *gérer les réservations clients,*
- *consulter les contrats,*
- *suivre la flotte de vélos.*

La base de données MySQL contient :

- *la liste des clients,*
- *les paiements,*
- *l'état de chaque vélo.*

⚠️ ***Incident critique***

À 9h32, un développeur stagiaire veut réinitialiser sa base de tests, mais au lieu d'exécuter sa commande sur sa base locale, il exécute :

```
1 DROP DATABASE velomax;
```

… sur la ***base de production***

En une seconde toutes les données de l'entreprise sont perdues.

ANALYSE DE L'INCIDENT

Problèmes identifiés

- *Pas de **sauvegarde récente** : la dernière sauvegarde date de 3 semaines.*
- *Le stagiaire avait tous les **droits** sur la base de **production**.*
- *Aucun **rôle** ou **privilège** n'était défini.*
- *Pas d'**environnement** de **test** distinct de la production.*

Conséquences

- *Site **indisponible** pendant 3 jours*
- ***Perte définitive** de 300 réservations*
- ***Réclamations** clients, demandes de **remboursement***
- *Atteinte à la **réputation** de l'entreprise*
- *L'équipe technique sous **pression**, le **stagiaire** en **détresse***

ENJEU DE CE COURS

Cet incident met en évidence deux failles majeures :

1. ***L'absence de stratégie de sauvegarde** :*
   - *Sans sauvegarde automatique ni politique de restauration, les données peuvent être perdues à jamais.*
   - *Sauvegarder régulièrement, automatiquement et tester les restaurations est vital.*

2. ***L'absence de contrôle des accès :***
   - *Tout utilisateur ne doit jamais avoir tous les droits.*
   - *Il faut :*
     - *Créer des utilisateurs distincts (admin, développeur, application...)*
     - *Définir les privilèges minimums*
     - *Utiliser les rôles MySQL pour regrouper et réutiliser les droits*

POURQUOI FAIRE DES SAUVEGARDES ?

1. Risques liés à la perte de données

Dans un système informatique, les données sont souvent le bien le plus précieux :

- Base **clients**, **commandes**, **paiements**, **configurations**, logs, etc.
- Une erreur humaine, une panne serveur ou un piratage peut **effacer** ou **corrompre** ces données.

Exemple concret : un développeur exécute un DELETE FROM sans clause WHERE → perte totale de la table.

👉 Sans sauvegarde, il est impossible de revenir en arrière.

Enjeu : savoir sauvegarder et restaurer efficacement les données est une compétence fondamentale pour tout professionnel du numérique.

POURQUOI FAIRE DES SAUVEGARDES ?

2. **Sécurité et organisation des accès**

- **Par défaut**, tous les **utilisateurs** ne doivent pas avoir **tous les droits** :
  - L'utilisateur d'une application web ne doit pas pouvoir supprimer des tables.
  - Un développeur junior ne doit pas pouvoir modifier des données de production.
- Il faut **structurer** l'**accès** en fonction des rôles :
  - **Lecture seule**
  - **Écriture limitée**
  - **Droits** d'**administration**
- MySQL offre des mécanismes de sécurité intégrés via les utilisateurs, privilèges et rôles.

Objectif : apprendre à créer des comptes, limiter les droits et assurer la sécurité des données.

POURQUOI FAIRE DES SAUVEGARDES ?

Prérequis à tout environnement de production

- En entreprise, on ne travaille **jamais sans stratégie de sauvegarde** ni contrôle des accès :
  - C'est un critère pour être en **conformité** avec le **RGPD**, les **normes ISO**, ou les exigences clients.

- Avoir des **sauvegardes régulières** et tester leur restauration fait partie des bonnes pratiques DevOps ou SysAdmin.

- De même, mettre en place une politique d'accès claire permet :
  - d'éviter les erreurs humaines,
  - de renforcer la sécurité contre les attaques internes ou externes,
  - de respecter le principe du moindre privilège.

COMPOSANTS DE MYSQL

Composant	Description
Serveur MySQL (mysqld)	Programme principal qui stocke les données, gère les connexions, les requêtes SQL, etc.
Client MySQL (mysql)	Interface en ligne de commande pour interagir avec le serveur
Base de données	Conteneur logique regroupant des tables
Table	Structure de données tabulaire stockant les lignes
Utilisateur	Entité disposant de droits pour accéder/modifier les données
Schéma	Autre nom pour une base de données dans MySQL
Moteur de stockage (InnoDB, MyISAM...)	Détermine comment les données sont physiquement stockées

OUTILS DISPONIBLES

Outil	**Rôle**
mysql	Client CLI pour se connecter et exécuter des requêtes
mysqldump	Outil de **sauvegarde/export** logique d'une base (format SQL)
mysqladmin	Outil d'administration rapide du serveur (arrêt, redémarrage, statistiques, etc.)

LES TYPES DE SAUVEGARDES

Type	Description	Exemple
Logique	Export des données et de la structure **sous forme de requêtes SQL** (fichier texte)	mysqldump
Physique	Copie brute des fichiers de données du serveur	Copie des dossiers /var/lib/mysql

Dans ce cours, on se concentre sur les **sauvegardes logiques** avec **mysqldump**

OUTIL MYSQLDUMP

Qu'est ce que mysqldump ?

- Utilitaire CLI fourni avec MySQL
- Permet d'exporter une base sous forme de requêtes SQL
- Le fichier généré est humainement lisible et modifiable

Commande de base

```
1 mysqldump -u user -p database_name > backup.sql
```

- -u user : nom de l'utilisateur MySQL
- -p : demande le mot de passe
- database_name : base à sauvegarder
- > : redirige la sortie dans un fichier
- backup.sql : fichier généré

EXEMPLE DE BACKUP.SQL

A noter

- /*!xxxx ... */ : ce sont des commentaires spéciaux exécutés uniquement par certaines versions de MySQL.
- Le fichier commence souvent par la configuration de l'encodage (utf8mb4).
- Chaque table est supprimée (DROP TABLE) puis recréée.
- Les données sont insérées avec des INSERT INTO.
- Les LOCK TABLES garantissent que les données ne changent pas pendant l'import.

```sql
-- MySQL dump 10.13  Distrib 8.0.33, for Linux (x86_64)
--
-- Host: localhost    Database: cineclub
-- ------------------------------------------------------
-- Server version       8.0.33

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Table structure for table `films`
--

DROP TABLE IF EXISTS `films`;
CREATE TABLE `films` (
  `id` int NOT NULL AUTO_INCREMENT,
  `titre` varchar(100) NOT NULL,
  `realisateur` varchar(100) NOT NULL,
  `annee` int NOT NULL,
  `duree` int DEFAULT NULL, -- en minutes
  `genre` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `films`
--

LOCK TABLES `films` WRITE;
/*!40000 ALTER TABLE `films` DISABLE KEYS */;
INSERT INTO `films` (`id`, `titre`, `realisateur`, `annee`, `duree`, `genre`)
VALUES
(1, 'Inception', 'Christopher Nolan', 2010, 148, 'Science-fiction'),
(2, 'Parasite', 'Bong Joon-ho', 2019, 132, 'Thriller'),
(3, 'La L
```

SAUVEGARDE COMPLETE

Une sauvegarde complète inclut :

- Les tables
- Les données
- Les vues
- Les triggers
- Les procédures stockées / fonctions

```
mysqldump -u user -p \
  --routines \
  --triggers \
  --single-transaction \
  --quick \
  database_name > full_backup.sql
```

Option	Rôle
--routines	Sauvegarde des fonctions et procédures stockées
--triggers	Inclut les triggers
--single-transaction	Sauvegarde cohérente sans verrouillage (utile avec InnoDB)
--quick	Lit les lignes une par une pour limiter l'usage mémoire

SAUVEGARDE PARTIELLE

Permet de ne sauvegarder que certaines tables, par exemple pour un export sélectif ou des tests

```
mysqldump -u user -p database_name table1 table2 > partielle.sql
```

Utile pour :

- sauvegarder une table de configuration
- isoler des données à transférer vers un autre système

Sauvegarde de la structure uniquement

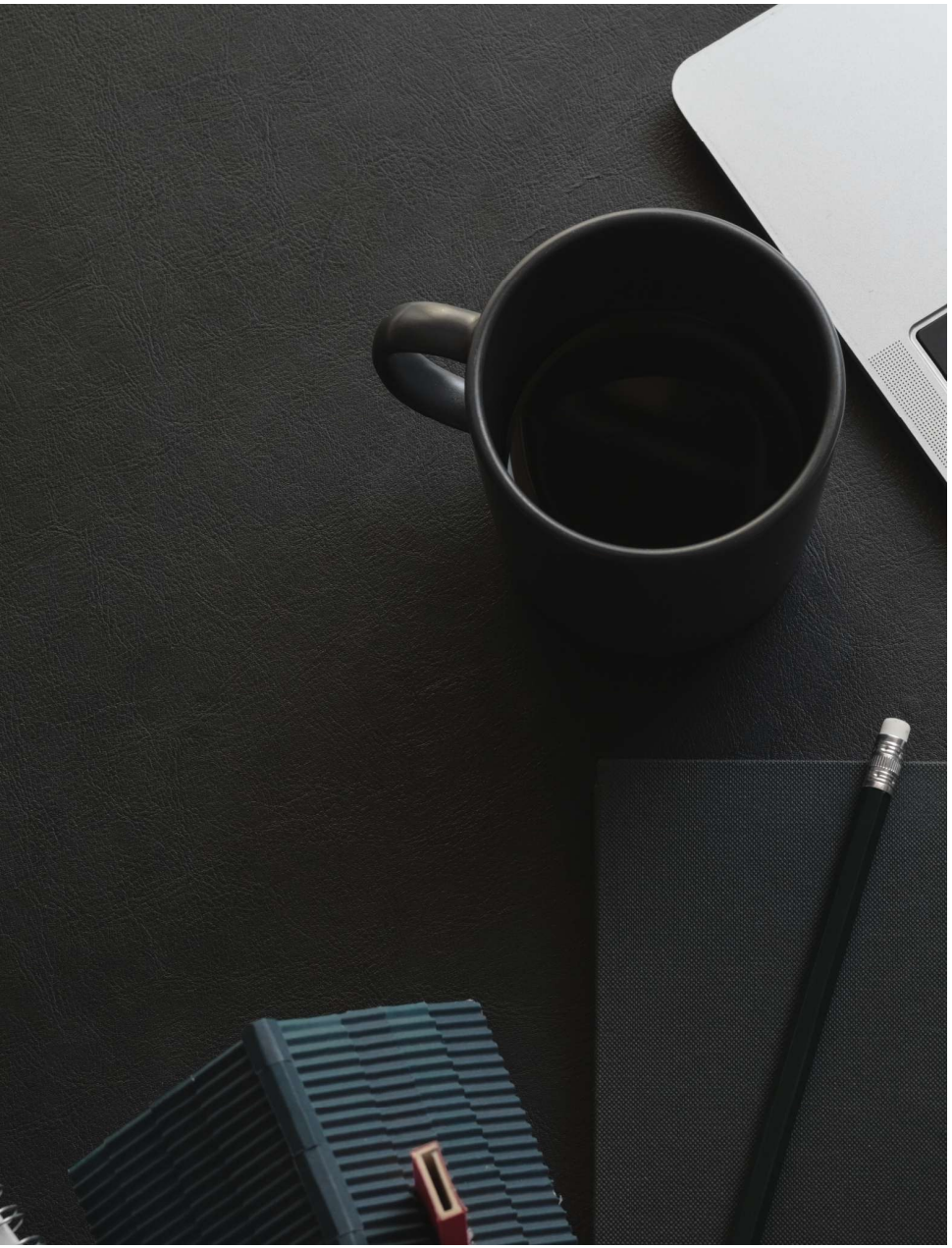
Sauvegarde du schéma sans les données (tables, colonnes, index, etc.).

```
mysqldump -u user -p --no-data database_name > structure.sql
```

Intérêts :
Reproduire la structure d'une base dans un autre environnement (ex : staging)
Partager un squelette de base avec des développeur

ATELIER

Réaliser les sauvegardes pour Cineclub



AUTOMATISATION DES SAUVEGARDES

SCRIPT BASH DE SAUVEGARDE

Créer un fichier backup.sh

```
#!/bin/bash

# Variables
USER="root"
DB="cineclub"
BACKUP_DIR="/c/Users/MonNom/Documents/backups"
DATE=$(date +"%Y%m%d-%H%M%S")

# Création du dossier si besoin
mkdir -p "$BACKUP_DIR"

# Exécution de la sauvegarde
mysqldump -u $USER -p $DB > "$BACKUP_DIR/backup-$DATE.sql"

echo "Sauvegarde terminée dans $BACKUP_DIR/backup-
$DATE.sql"
```

Donnez les droits d'execution dans le GitBash

```
chmod +x backup.sh
```

Executer le script

```
./backup.sh
```

PLANNIFIER LA TACHE

Avant de plannifier la tâche on se rend compte pour le moment que le script nous demande d'écrire notre mot de passe.

Si on souhaite plannifier la tâche, il n'y aura plus d'interaction avec l'utilisateur. Il faut donc gérer le mot de passe de façon différente.

En travaillant sur Linux nous pouvons configurer un fichier .my.cnf au niveau de l'utilisateur 'root' qui sera lu automatiquement par sql. C'est la solution la plus utilisée car les serveurs sont sur des distributions Linux.

Pour notre part et pour tester sur notre OS, nous allons utiliser mysql_config_editor

exemple de fichier .my.cnf

```
[client]
user=root
password=MonSuperMotDePasse
```

Attention ce fichier ne doit être lisible que par toi :

```
chmod 600 ~/.my.cnf
```

Cela permet de donner des droits d'accès très limités à ce fichier. Il n'y a que le propriétaire qui peut lire et écrire.

MYSQL_CONFIG_EDITOR

1. Ouvrir l'invite de commande cmd

> mysql_config_editor --help

- Si tu vois la liste des options, c'est bon, l'outil est disponible.
- Sinon, il faut ajouter le dossier bin de MySQL à ta variable d'environnement PATH.

2. Créer un profil de connexion sécurisé

> mysql_config_editor set --login-path=local --host=localhost --user=root --password

- —login-path=local : nom du profil (tu peux choisir un autre nom)
- —host=localhost : adresse du serveur MySQL (souvent localhost)
- —user=root : ton utilisateur MySQL
- —password : le programme te demandera ensuite ton mot de passe

MYSQL_CONFIG_EDITOR

3. Entrer le mot de passe

```
Enter password: ********
```

Tape ton mot de passe MySQL (tu ne verras rien s'afficher, c'est normal) et appuie sur Entrée

4. Vérifier que ton profil a bien été créé

> mysql_config_editor print --login-path=local

```
[local]
user = root
host = localhost
```

Le mot de passe n'est pas affiché pour des raisons de sécurité

5. Utiliser ce profil dans tes commandes mySQL

À partir de maintenant, tu peux te connecter à MySQL sans taper le mot de passe

> mysql --login-path=local

Ou faire un dump

> mysqldump --login-path=local nom_de_la_base > backup.sql

On peut désormais adapter notre script comme suit :

```bash
#!/bin/bash

DATE=$(date +'%Y%m%d-%H%M%S')
BACKUP_DIR="/c/Users/TonNom/backups"
mkdir -p "$BACKUP_DIR"

mysqldump --login-path=local cineclub > "$BACKUP_DIR/backup-$DATE.sql"

```

CRÉER UNE TACHE PLANIFIEE

1. Préparer son script

On vérifie que son script est bien .sh backup_mysql.sh

2. Créer une tâche avec le plannificateur

- Dans le volet de droite Créer une tâche
- Donne un nom à ta tâche par exemple Backup MySQL
- Optionnel ajoute une description
- Selectionner Executer même si l'utilisateur n'est pas connecté
- Cochez Executez avec les autorisations maximales

3. Ajouter un Déclencheur

- Va dans l'onglet Déclencheur
- Clique sur nouveau
- Paramètre la tache (début, répétition)
- clicke sur OK

3. Onglet Actions

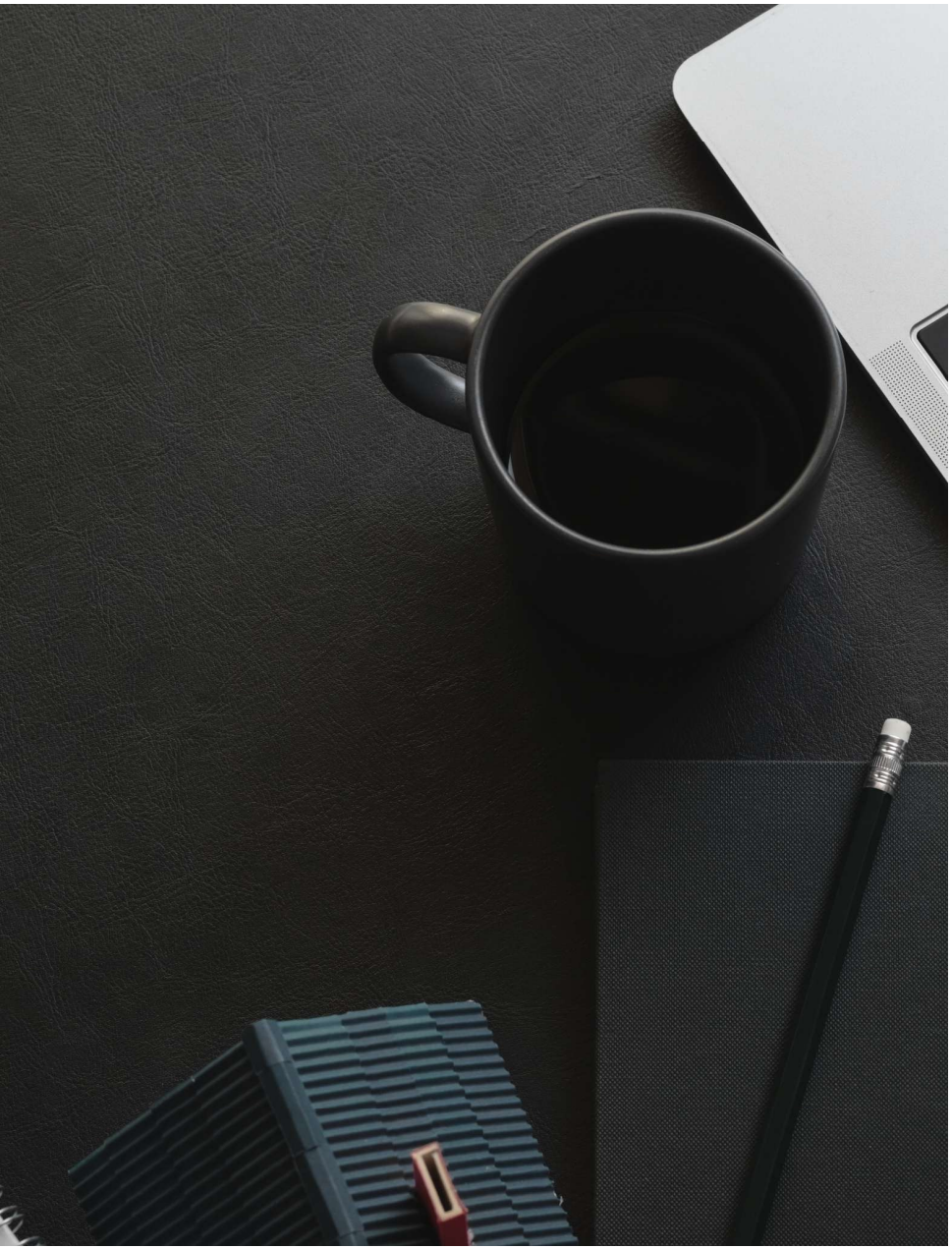
- Va dans l'onglet Actions
- Selectionne Démarrer un programme
- Dans Programmes/Script :
  - C:\Program Files\Git\bin\bash.exe
- Dans ajouter des arguments indique le script :
  - /c/Users/TonNom/backup_mysql.sh

4. Valider et tester

- Tu peux faire un click droit sur la tâche dans la liste → Exécuter pour tester

ATELIER

Faire la tache plannifiée pour Cineclub



RESTAURATION DE LA BASE

COMMANDE DE BASE

La restauration d'une base MySQL à partir d'un fichier de sauvegarde .sql s'effectue généralement avec la commande suivante :

```
mysql -u user -p database_name < backup.sql
```

- mysql : le client MySQL pour exécuter des commandes SQL.
- -u user : spécifie l'utilisateur MySQL.
- -p : demande le mot de passe (tu seras invité à le saisir).
- database_name : nom de la base dans laquelle importer les données.
- < backup.sql : redirection du fichier SQL en entrée, qui contient les instructions à exécuter (création de tables, insertion de données, etc.).

RESTAURATION DANS UNE NOUVELLE BASE

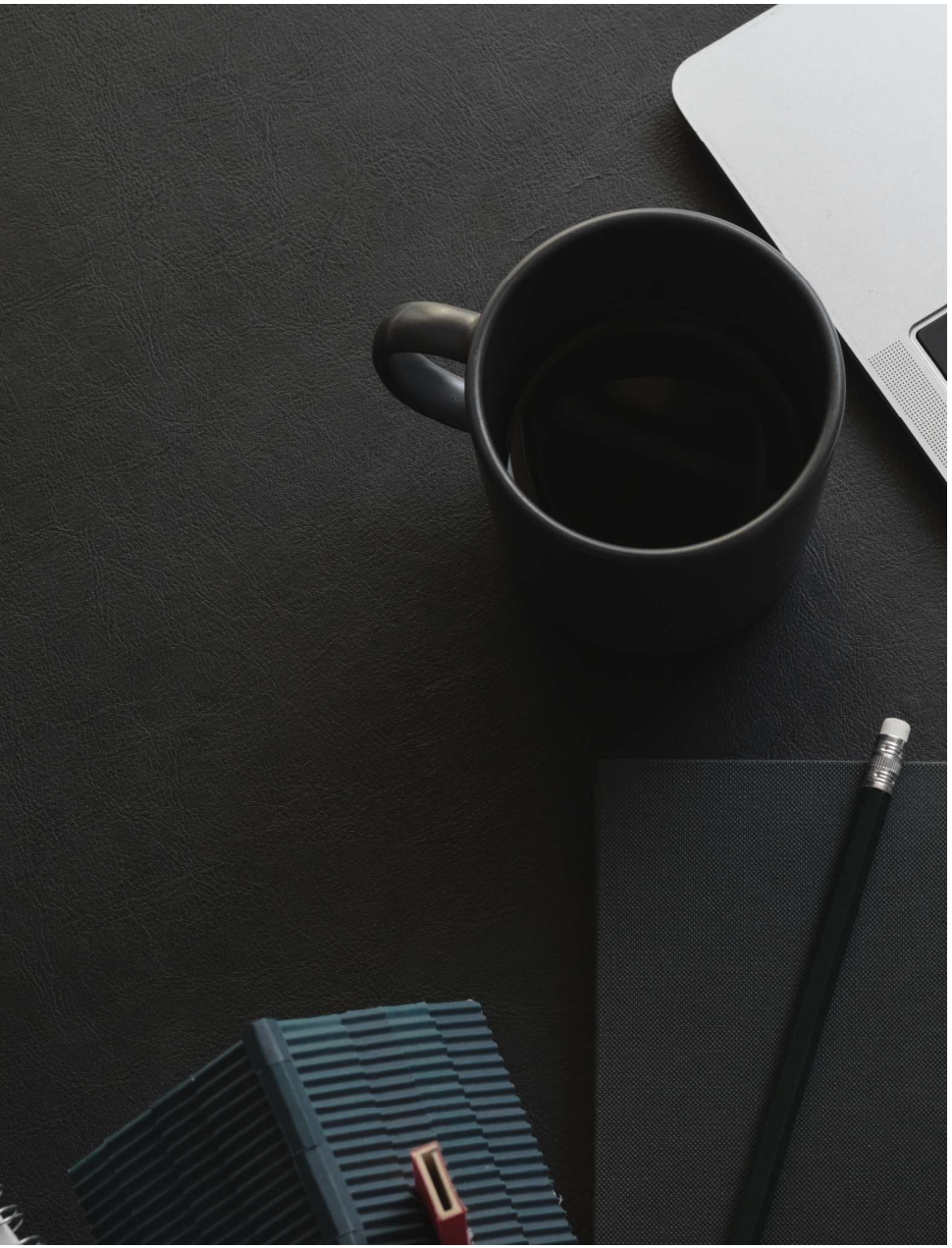
Avant d'importer la sauvegarde dans une base nouvelle, il faut créer cette base :

```
1 CREATE DATABASE nouvelle_base CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

CHARACTER SET et COLLATE garantissent la bonne gestion des caractères, important notamment si la sauvegarde contient des données en UTF-

Une fois la nouvelle base créée tu peux lancer

```
1 mysql -u user -p database_name < backup.sql
```



GESTION DES UTILISATEURS ET DES ROLES

UTILISATEURS MYSQL

Les utilisateurs MySQL représentent les entités qui peuvent se connecter au serveur MySQL pour accéder aux bases et effectuer des actions selon les droits qui leur sont accordé

Créer un utilisateur

```
1 CREATE USER 'jean'@'localhost' IDENTIFIED BY 'motdepasse';
```

- 'jean' : nom de l'utilisateur.
- 'localhost' : origine depuis laquelle l'utilisateur peut se connecter (ici, uniquement depuis la machine locale).
- 'motdepasse' : mot de passe associé à cet utilisateur.

Remarques :

- Pour autoriser une connexion depuis n'importe quelle machine, on peut remplacer 'localhost' par '%'.
- L'utilisateur est créé sans privilèges par défaut. Les droits sont donnés ensuite avec GRANT

UTILISATEURS MYSQL

Supprimer un utilisateur

```
1 DROP USER 'jean'@'localhost';
```

- Supprime l'utilisateur de la base de données des utilisateurs.
- Ne laisse plus aucun accès au serveur MySQL à cet utilisateur

PRIVILEGES

Les privilèges définissent ce qu'un utilisateur peut ou ne peut pas faire sur le serveur MySQL.

Accorder des privilèges

```
1 GRANT SELECT, INSERT ON entreprise.* TO 'jean'@'localhost';
```

- SELECT, INSERT : les types d'opérations autorisées.
- entreprise.* : la base de données entreprise et toutes ses tables (*).
- 'jean'@'localhost' : utilisateur ciblé.

Cela donne donc à Jean le droit de lire (SELECT) et d'insérer (INSERT) des données dans toutes les tables de la base entreprise

PRIVILEGES

Révoquer des privilèges

```
1 REVOKE INSERT ON entreprise.* FROM 'jean'@'localhost';
```

- Retire le droit d'INSERT à l'utilisateur jean sur la base entreprise.
- Il conservera les autres droits (comme SELECT) si existants

Voir les droits d'un utilisateur

```
1 SHOW GRANTS FOR 'jean'@'localhost';
2
```

Revoquer tous les privilèges

```
1 REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'jean'@'localhost';
```

- Retire tous les droits ainsi que la possibilité d'accorder des droits à d'autres (GRANT OPTION).
- Utile avant de supprimer un utilisateur ou pour réinitialiser ses droits.

ROLES (MYSQL 8+)

Les rôles sont des ensembles de privilèges qu'on peut attribuer à un ou plusieurs utilisateurs pour simplifier la gestion des droits

Créer un rôle

```
1 CREATE ROLE 'lecteur';
```

- Crée un rôle nommé lecteur.
- Par défaut, un rôle ne peut rien faire avant attribution des droits

Attribuer des droits à un rôle

```
1 GRANT SELECT ON entreprise.* TO 'lecteur';
```

Le rôle lecteur obtient le droit de lecture sur toutes les tables de la base entreprise

ROLES (MYSQL 8+)

Assigner un rôle à un utilisateur

```
1 GRANT 'lecteur' TO 'jean'@'localhost';
```

- L'utilisateur jean se voit attribuer le rôle lecteur.
- Ce rôle regroupe les droits configurés précédemment

Assigner un rôle par défaut

```
1 SET DEFAULT ROLE 'lecteur' TO 'jean'@'localhost';
```

- Quand jean se connecte, le rôle lecteur est automatiquement activé.
- Sinon, un utilisateur peut activer ou désactiver un rôle via :

```
1 SET ROLE 'lecteur';
```

TABLEAU RECAPITULATIF

Action	Commande SQL	But
Créer utilisateur	CREATE USER 'user'@'host' IDENTIFIED BY 'pwd';	Créer un compte sans droits initiaux
Supprimer utilisateur	DROP USER 'user'@'host';	Supprimer un compte
Accorder droits	GRANT SELECT ON db.* TO 'user'@'host';	Donner des privilèges précis
Révoquer droits	REVOKE INSERT ON db.* FROM 'user'@'host';	Retirer certains privilèges
Voir droits	SHOW GRANTS FOR 'user'@'host';	Vérifier les droits d'un utilisateur
Créer rôle	CREATE ROLE 'role';	Créer un ensemble de privilèges
Donner droits au rôle	GRANT SELECT ON db.* TO 'role';	Définir droits sur un rôle
Assigner rôle	GRANT 'role' TO 'user'@'host';	Attribuer un rôle à un utilisateur
Activer rôle par défaut	SET DEFAULT ROLE 'role' TO 'user'@'host';	Auto-activation du rôle à la connexion

ATELIER

Gérer les utilisateurs et les rôles sur cineclub

CONSIGNES

Étape 1 : Créer des utilisateurs

1. Connectez-vous à MySQL en tant qu'utilisateur root ou administrateur.
2. Créez les utilisateurs suivants :

- admin avec le mot de passe admin123 (accès local uniquement)
- lecteur avec le mot de passe lecture456 (accès local uniquement)
- gestionnaire avec le mot de passe gest789 (accès local uniquement)

Étape 2 : Gérer les privilèges individuels

- Attribuez à l'utilisateur admin tous les privilèges sur la base cineclub.
- Donnez à l'utilisateur lecteur seulement le droit de lire (SELECT) toutes les tables de la base cineclub.
- Donnez à l'utilisateur gestionnaire le droit d'insérer (INSERT), modifier (UPDATE) et supprimer (DELETE) dans les tables films uniquement.
- Vérifiez les privilèges de chaque utilisateur avec SHOW GRANTS

CONSIGNES

Étape 3 : Utiliser des rôles pour simplifier la gestion

1. Créez deux rôles :
   - role_lecteur avec le droit SELECT sur toute la base cineclub.
   - role_gestionnaire avec les droits INSERT, UPDATE, DELETE sur les tables films et projections.
2. Assignez ces rôles aux utilisateurs correspondants (lecteur et gestionnaire).
3. Activez le rôle par défaut pour chaque utilisateur.
4. Vérifiez les rôles et privilèges des utilisateurs.

Étape 4 : Tester les accès

- Connectez-vous à MySQL en tant que lecteur et essayez de :
  - Lister les films (requête SELECT).
  - Modifier un film (requête UPDATE) – cela doit être refusé.
- Connectez-vous en tant que gestionnaire et essayez de :
  - Ajouter un nouveau film (requête INSERT).
  - Supprimer une projection (requête DELETE).
  - Lister tous les abonnés (requête SELECT) – cela doit être refusé.
- Connectez-vous en tant que admin et vérifiez que vous pouvez tout faire.

QUIZZ