Projet Spicy Invader

UX, OO, DB

Ethan Schafstall

28.08.2023-03.11.2023

Table des matières

[Expérience utilisateur (UX) 1](#_Toc147912512)

[Introduction 1](#_Toc147912513)

[Analyse 1](#_Toc147912514)

[Conception 1](#_Toc147912515)

[Évaluation 1](#_Toc147912516)

[Programmation orientée objet (OO) 1](#_Toc147912517)

[Introduction 1](#_Toc147912518)

[Analyse fonctionnelle 1](#_Toc147912519)

[Analyse technique 1](#_Toc147912520)

[Tests Unitaire 1](#_Toc147912521)

[ChatGPT 1](#_Toc147912522)

[Conclusion 1](#_Toc147912523)

[Base de données (DB) 1](#_Toc147912524)

[Introduction 1](#_Toc147912525)

[Importation données et schéma de base de données 1](#_Toc147912526)

[Gestions des utilisateurs 1](#_Toc147912527)

[Requêtes de sélection 3](#_Toc147912528)

[Création des index 6](#_Toc147912529)

[Backup/Restore 6](#_Toc147912530)

# Expérience utilisateur (UX)

## Introduction

## Analyse

## Conception

## Évaluation

# Programmation orientée objet (OO)

## Introduction

## Analyse fonctionnelle

## Analyse technique

## Tests Unitaire

## ChatGPT

## Conclusion

# Base de données (DB)

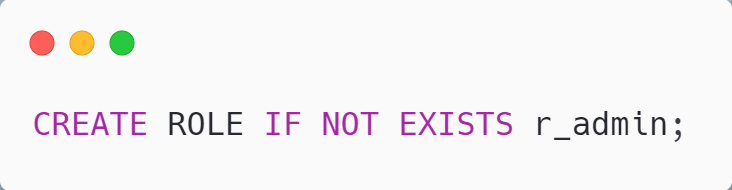
# Introduction

## Importation données et schéma de base de données

## Gestions des utilisateurs

### Administrateur de jeu

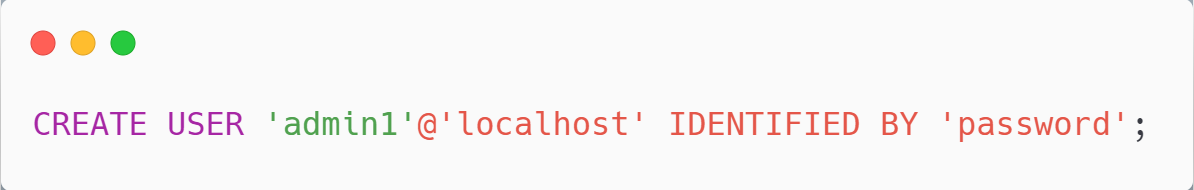
Création du rôle r\_admin avec le command **CREATE**.



Donne les privilèges au rôle r\_admin avec command **GRANT**. **WITH GRANT OPTION** et **\*.\*** pour donner tous les privilèges au rôle, donc créer un rôle "super user"



La création d’un user "admin" avec **CREATE USER**, nommée admin1

****

L’ajoute du user admin1 au rôle r\_admin avec la commande **GRANT ROLE**.

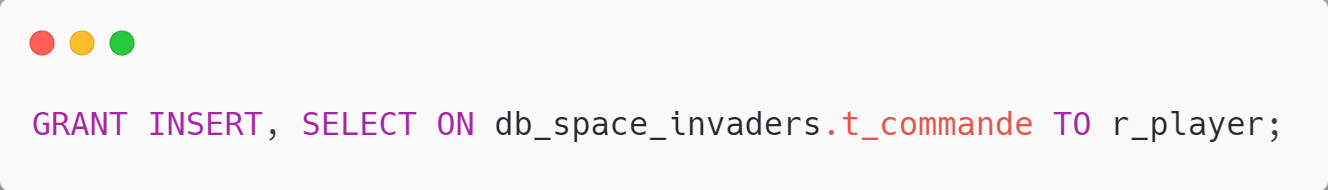
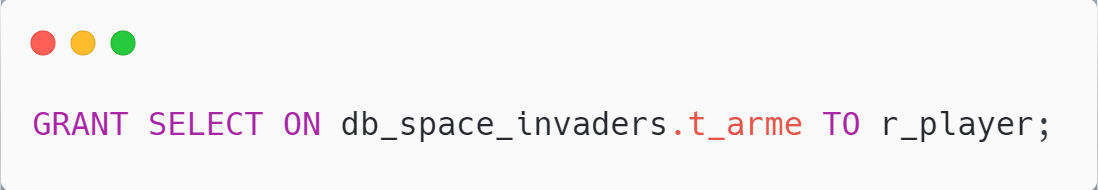


### Joueur

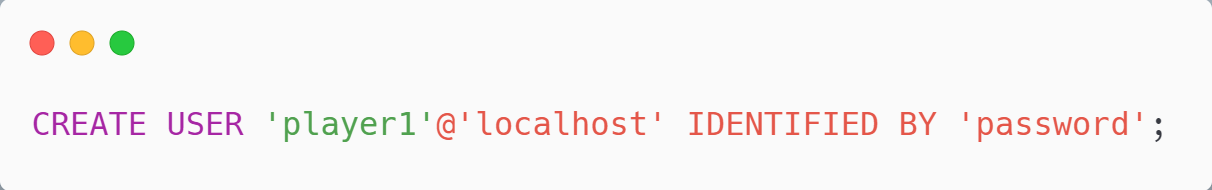
Création du rôle r\_player avec le command **CREATE**.



Donne les privilèges au rôle r\_player avec command **GRANT** **SELECT ON db\_space\_invaders.t\_arme** car le joueur n’a que le droit de récupérer des données sur les armes. **GRANT** **INSERT, SELECT ON db\_space\_invaders.t\_commade** car le jour à le droit d’insérer, et récupérer des donnes dans la table commande. Pour que le joueur puisse voir les infos sur les armes, leurs prix, etc. Et ensuite effectuer des commandes.



La création d’un user "joueur" avec **CREATE USER**, nommée player1

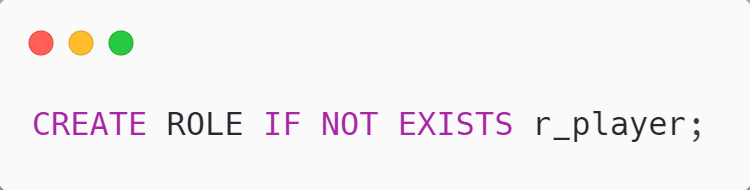


L’ajoute du user player1 au rôle r\_player avec la commande **GRANT ROLE**.

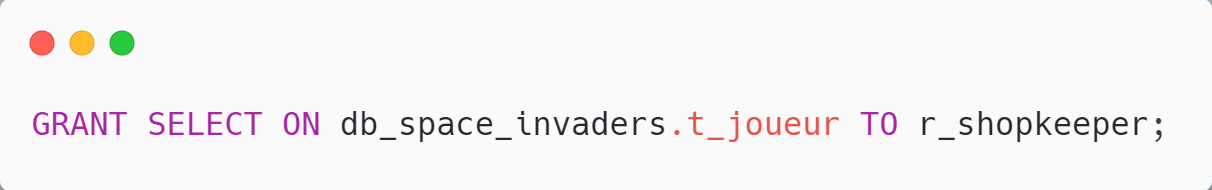


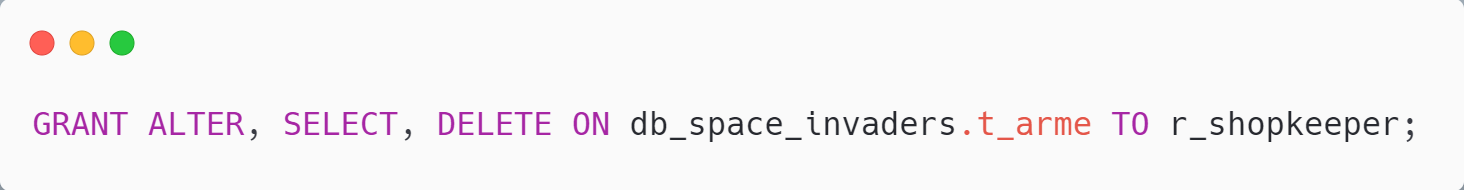
### Gestionnaire de la boutique

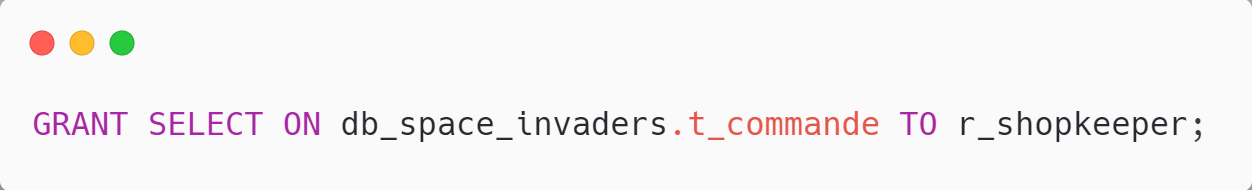
Création du rôle r\_shopkeeper avec le command **CREATE**.



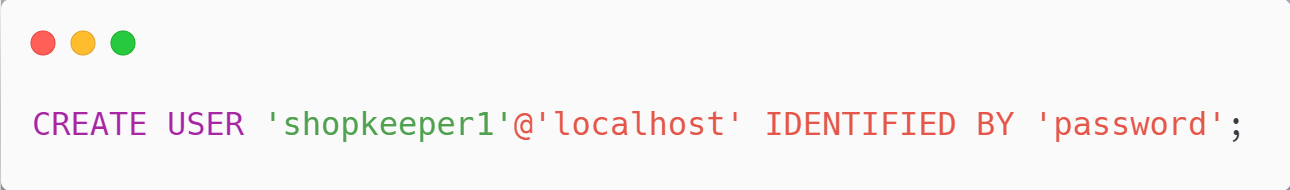
Donne les privilèges au rôle r\_shopkeeper avec command **GRANT** **SELECT ON db\_space\_invaders.t\_joueur** car le shopkeeper n’a que le droit de récupérer des données sur les joueurs. **GRANT** **ALTER, SELECT, DELETE ON db\_space\_invaders.t\_arme** pour pouvoir ajouter de nouvelles armes/modifier les prix. Et **GRANT SELECT ON db\_space\_invaders.t\_commade** pour voire toutes commandes faites.



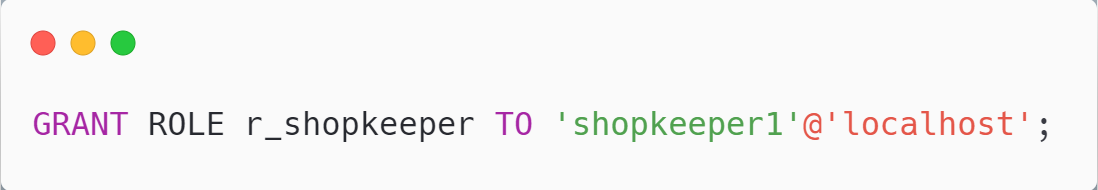




La création d’un user "shopkeeper" avec **CREATE USER**, nommée shopkeeper1



L’ajoute du user shopkeeper1 au rôle r\_shopkeeper avec la commande **GRANT ROLE**.



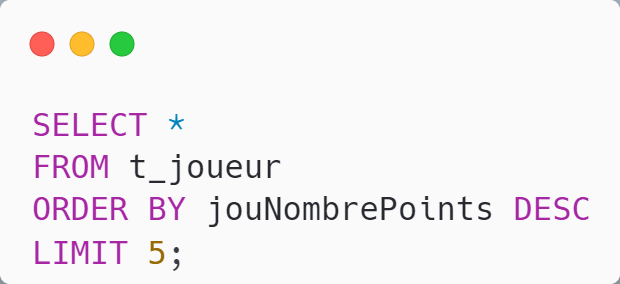
## Requêtes de sélection

Les requêtes de select SQL servent à récupérer des donnés depuis la base de données du jeu, pour ensuite afficher/ou en servir pour l’UX, ainsi que le fonctionnement du jeu.

### Requête n°1

Cette requête récupère toutes les colonnes de **t\_joueur**, les ordonner parce leur nombre de points en décroissant, et limitée à cinq.

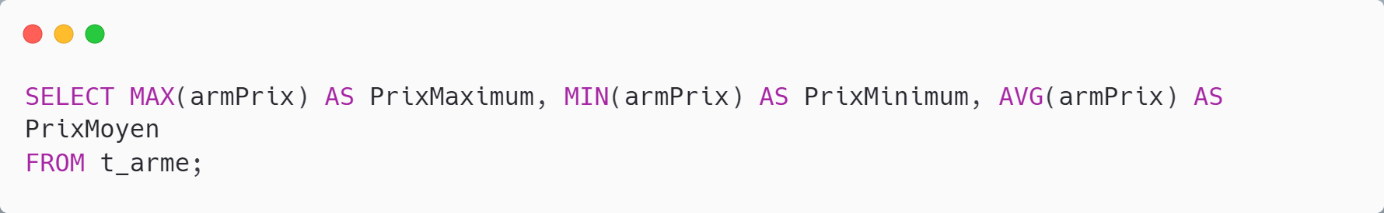
Donc elle montre les top cinq joueurs parce nombre de point décroissant va être afficher.



### Requête n°2

Cette requête récupère le **MAX**, **MIN**, et **AVG** du tableaux **t\_arme**.

Donc elle récupère l’arme qui coute le plus cher, le moins cher, et le prix moyenne d’une arme.



### Requête n°3

Cette requête récupère le **COUNT** de **idCommande** et le **fkJoueur** du tableaux **t\_idCommande**, groupé par le **fkJoueur**, et ordonner par le **COUNT** de **idCommande DESC.**

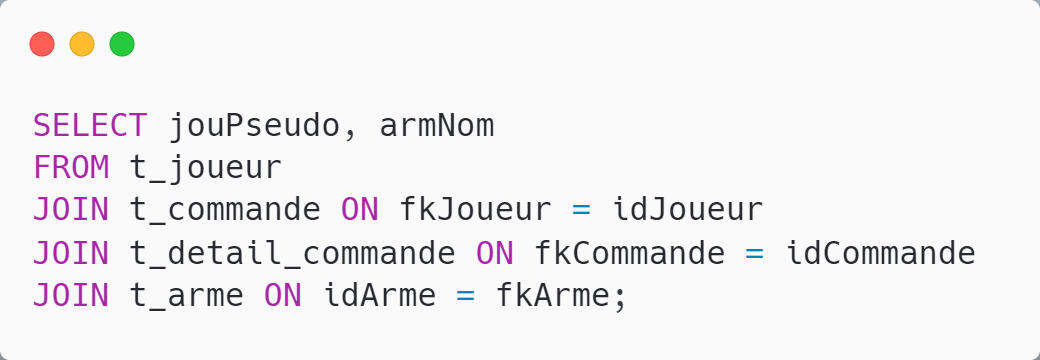
Donc elle récupère le id de chaque joueur et le nombre de commandes qu’ils ont effecteur, ordonner par le **COUNT DESC** pour les organiser par du joueur avec les plus au jouer avec le moins de commandes.



### Requête n°4

Cette requête récupère le **COUNT** **idCommande** et le **fkJoueur** du tableaux **t\_commande**, groupé par le **fkJoueur**, et groupée par le fkJoueur **HAVING** **COUNT** de **idCommande** plus que 2.

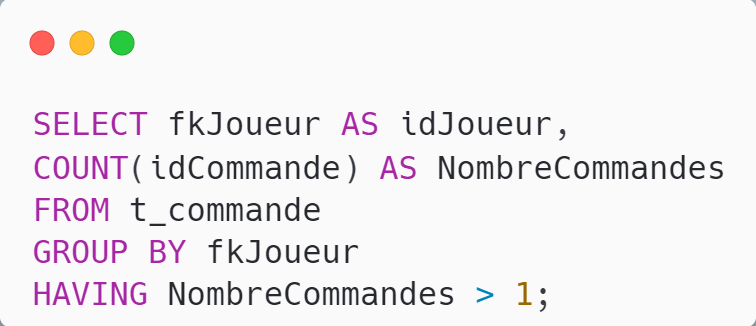
Seulement les joueurs qui ont passé plus que 2 commandes vont être afficher.



### Requête n°5

Cette requête récupère le **jouPseudo** de **t\_joueur**, le **armNom** de **t\_arme**, et comNumeroCommade de **t\_joueur.** Des **JOIN** sont utiliser pour faire les liaisons entre les différents tableux à travers de id et fk.

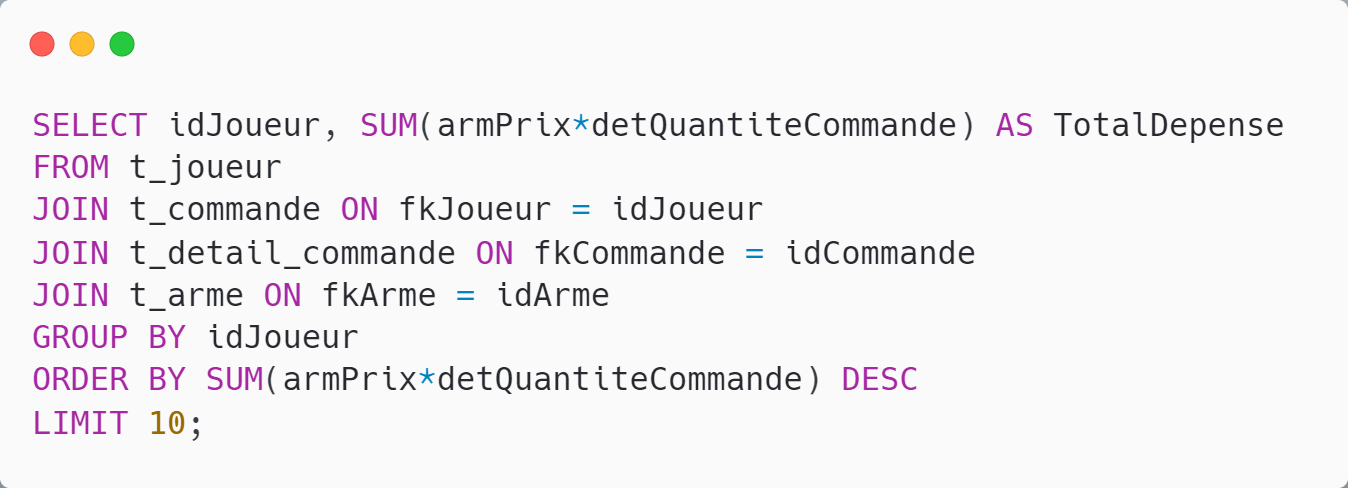
Cette requête affiche les pseudos et les armes qu’ils ont acheté dans chaque commande.



### Requête n°6

Cette requête récupère le **idJoueur** de **t\_joueur**, le **SUM armPrix \* detQuantiteCommande** de **t\_arme** et **t\_detail\_commande.** Des **JOIN** sont utiliser pour faire les liaisons entre les différents tableux à travers de id et fk. C’est groupé par le **idJoueur**, ordonnée par **SUM armPrix \* detQuantiteCommande DESC**, et **LIMIT** 10.

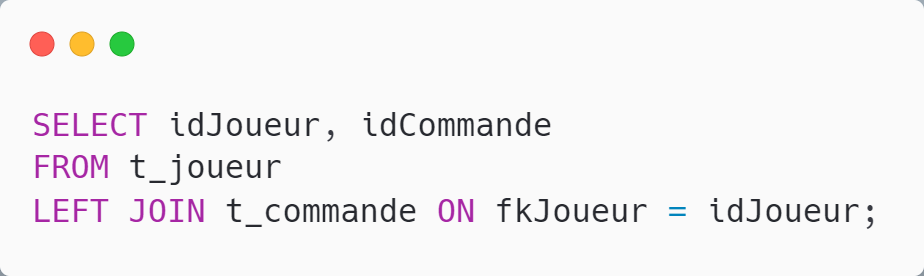
Cette requête affiche les identifient des joueurs et le total qu’ils ont dépensé, avec seulement le top 10 qui ont dépensé le plus en ordre décroissant.



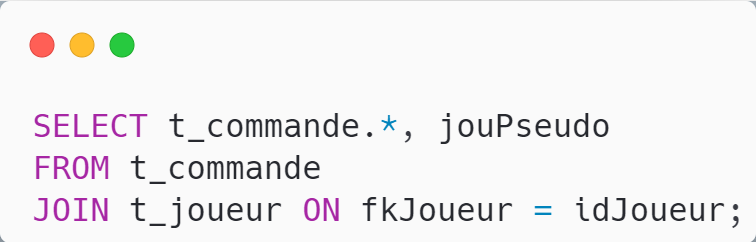
### Requête n°7

Cette requête récupère **idJoueur** de **t\_joueur**, et le **idCommande** de **t\_commande**. **t\_commande** est relier à **t\_joueur** via un **LEFT JOIN**.

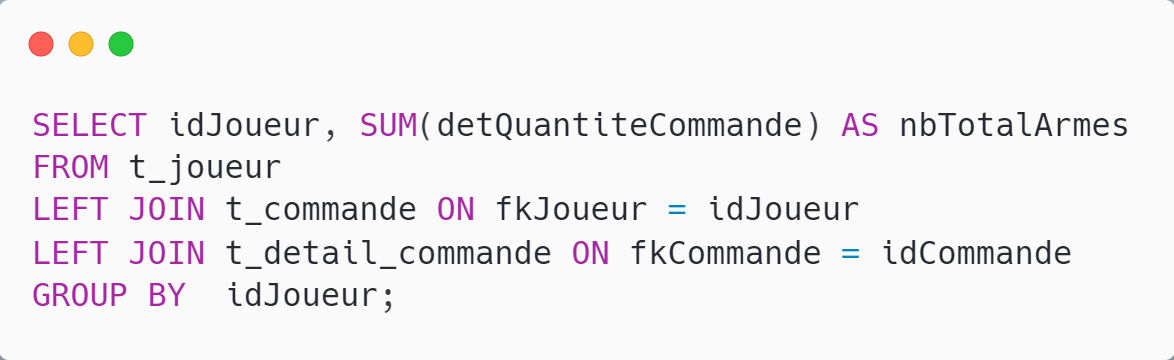
Cette requête affiche tous les joueurs, et leurs commandes, ainsi que ceux qui n’ont pas passé de commandes.



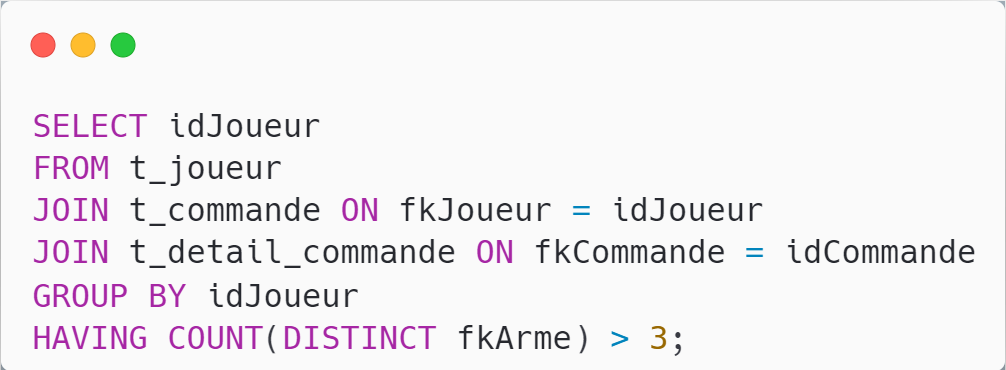
### Requête n°8



### Requête n°9



### Requête n°10



## Création des index

### 3.5.1.

Car MySQL va automatiquement créer des indexes sur les colonnes uniques, les clés primaires, et les clés étrangères.

### 3.5.2.

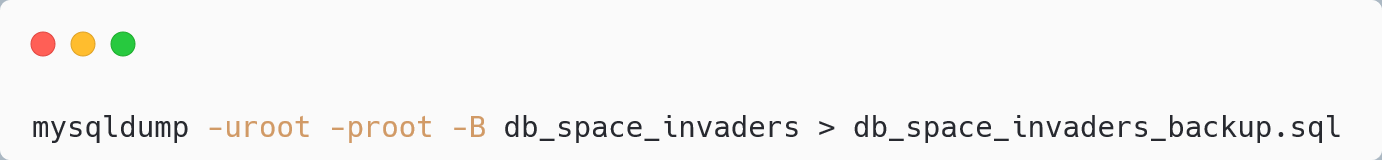
Les index vont permettre d’effectuer plus rapidement les requêtes

### 3.5.3.

Ça serait pertinent de mettre un index sur le champ 'comDate' de la table 't\_commande' car lorsqu'une requête vise à recouper des données en fonction de ce champ (avant une certaine date, entre deux dates spécifiques), cela peut prendre du temps. Par exemple, lorsqu'on veut récupérer toutes les commandes effectuées avant le 31 décembre 2023, MySQL devra parcourir tous les jours de tous les mois avant d'atteindre la date du 31 décembre 2023. Ce n'est pas très efficace et cela prendrait plus de temps que nécessaire.

## Backup/Restore

### Backup



### Restore

