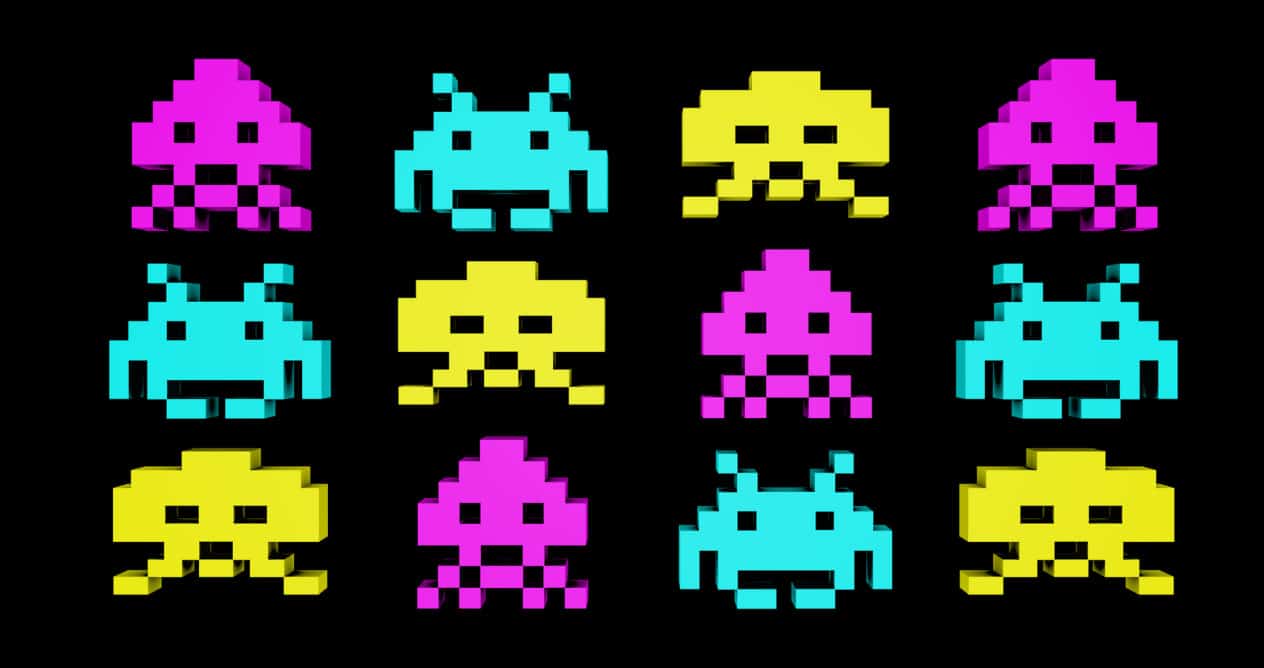
DB, SQL – Spicy Invaders



LEBET Esteban – CID2A

ETML, Vennes

24 Périodes

CHARMIER Grégory

[Lien vers Cahier des charges](https://eduvaud-my.sharepoint.com/personal/pn23stc_eduvaud_ch/Documents/2.%20DEUXIEME%20ANNÉE%20(CID2A)/P-DB-GCR-SpicyInvaders/P_OO-Specifications.pdf)

Table des matières

[DB, SQL – Spicy Invaders 1](#_Toc149918044)

[1. Introduction 3](#_Toc149918045)

[2. Gestion des utilisateurs 3](#_Toc149918046)

[2.1. Administrateur de jeu 3](#_Toc149918047)

[2.2. Joueur 3](#_Toc149918048)

[2.3. Gestionnaire de la boutique 4](#_Toc149918049)

[3. Requêtes SQL 4](#_Toc149918050)

[3.1. Requête 1 4](#_Toc149918051)

[3.2. Requête 2 4](#_Toc149918052)

[3.3. Requête 3 5](#_Toc149918053)

[3.4. Requête 4 5](#_Toc149918054)

[3.5. Requête 5 5](#_Toc149918055)

[3.6. Requête 6 6](#_Toc149918056)

[3.7. Requête 7 6](#_Toc149918057)

[3.8. Requête 8 6](#_Toc149918058)

[3.9. Requête 9 7](#_Toc149918059)

[3.10. Requête 10 7](#_Toc149918060)

[4. Index 8](#_Toc149918061)

[4.1. Pourquoi certains index existent déjà 8](#_Toc149918062)

[4.2. Avantages et désavantages des Index 8](#_Toc149918063)

[4.2.1. Avantages 8](#_Toc149918064)

[4.2.2. Désavantages 8](#_Toc149918065)

[4.3. Où ajouter des Index pertinents 8](#_Toc149918066)

[5. Backup / Restore 8](#_Toc149918067)

[5.1. Backup 8](#_Toc149918068)

[5.2. Restore 8](#_Toc149918069)

# Introduction

J’ai été chargé de la réalisation d’un projet s’appelant « Spicy Invaders », un projet qui se divise sur trois aspects de l’informatique, La POO (programmation orientée objet), L’UX (user experience) et une dernière partie SQL, DB. C’est de cette partie que ce document traite.

# Gestion des utilisateurs

## Administrateur de jeu

L’administrateur de jeu peut créer, lire, mettre à jour et supprimer n’importe quelle table.

Il peut aussi gérer les utilisateurs et leurs privilèges.

Voici les différentes commandes à effectuer pour lui octroyer les droits dont il a besoin :

* **Lui créer un utilisateur sur la base de données.**

CREATE USER ‘**exempleAdmin’**@’localhost’ IDENTIFIED BY ‘**exempleMdp**’;

* **Lui créer un rôle sur la base de données**

CREATE ROLE ‘**administrateur**’ ;

* **Donner les différents privilèges à ce rôle**

GRANT ALL PRIVILEGES ON **db\_spaceinvaders.**\* TO ‘**admin’** WITH GRANT OPTION;

* **Lui attribuer le rôle créé**

GRANT ‘**admin** TO ‘**exempleAdmin’**@’**localhost**’

## Joueur

Le joueur peut lire les informations des armes (pour voir quelles armes il peut acheter).

Il peut également créer une commande. Finalement il peut aussi lire toutes les commandes.

Voici les différentes commandes à effectuer pour lui octroyer les droits dont il a besoin :

* **Lui créer un utilisateur sur la base de données.**

CREATE USER ‘**exempleJoueur’**@’localhost’ IDENTIFIED BY ‘**exempleMdp’ ;**

* **Lui créer un rôle sur la base de données**

CREATE ROLE ‘**joueur**’ ;

* **Donner les différents privilèges à ce rôle**

GRANT SELECT ON **armes** TO **joueur** ;

* **Lui attribuer le rôle créé**

GRANT ‘**joueur’** TO ‘**exempleJoueur’**@’**localhost**’

## Gestionnaire de la boutique

Le gestionnaire de la boutique peut lire les informations sur tous les joueurs (pour savoir qui a passé une commande), mettre à jour lire et supprimer des armes (ajout de nouvelles armes, modification des prix, etc.) et finalement il peut lire toutes les commandes.

Voici les différentes commandes à effectuer pour lui octroyer les droits dont il a besoin :

* **Lui créer un utilisateur sur la base de données.**

CREATE USER ‘**exempleGestionnaire’**@’localhost’ IDENTIFIED BY ‘**exempleMdp**’;

* **Lui créer un rôle sur la base de données**

CREATE ROLE ‘**gestionnaire**’ ;

* **Donner les différents privilèges à ce rôle**

GRANT SELECT ON **joueurs** TO **gestionnaire**;

GRANT UPDATE, SELECT, DELETE ON **armes** TO **gestionnaire**;

GRANT SELECT ON **commandes** TO **gestionnaire**;

* **Lui attribuer le rôle créé**

GRANT ‘**gestionnaire** ON ‘**exempleGestionnaire’**@**’localhost’**

# Requêtes SQL

## Requête 1

***La première requête que l’on vous demande de réaliser est de sélectionner les 5 joueurs qui ont le meilleur score c’est-à-dire qui ont le nombre de points le plus élevé. Les joueurs doivent être classés dans l’ordre décroissant :***

**SELECT \* FROM t\_joueur ORDER BY jouNombrePoints DESC LIMIT 5 ;**

Cette requête sélectionne les 5 joueurs ayant le score le plus élevé (le nombre de points le plus élevé) en ordre décroissant. Elle renvoie toutes les colonnes de la table **t\_joueur** pour ces joueurs.

## Requête 2

**Trouver le prix maximum, minimum et moyen des armes. Les colonnes doivent avoir pour nom « PrixMaximum », « PrixMinimum » et « PrixMoyen)**

**SELECT MAX(armPrix) AS PrixMaximum, AVG(armPrix) AS PrixMoyen,**

**MIN(armPrix) AS PrixMinimum FROM t\_arme ;**

Cette requête calcule le prix maximum, minimum et moyen des armes. Les résultats sont renommés en tant que "PrixMaximum," "PrixMinimum," et "PrixMoyen" à l'aide d'alias.

## Requête 3

***Trouver le nombre total de commandes par joueur et trier du plus grand nombre au plus petit.***

***La 1ère colonne aura pour nom "IdJoueur", la 2ème colonne aura pour nom "NombreCommandes"***

**SELECT fkJoueur AS IdJoueur, COUNT(fkJoueur) AS NombreCommandes**

**FROM t\_commande**

**GROUP BY fkJoueur**

**ORDER BY NombreCommandes DESC ;**

Cette requête trouve le nombre total de commandes passées par chaque joueur, en triant du joueur avec le plus grand nombre de commandes au joueur avec le moins de commandes. Les colonnes de résultats sont nommées "**IdJoueur**" et "**NombreCommandes**."

## Requête 4

***Trouver les joueurs qui ont passé plus de 2 commandes. La 1ère colonne aura pour nom "IdJoueur", la 2ème colonne aura pour nom "NombreCommandes"***

**SELECT fkJoueur AS IdJoueur, COUNT(fkJoueur) AS NombreCommandes**

**FROM t\_commande**

**GROUP BY fkJoueur**

**HAVING COUNT(fkJoueur) > 2 ;**

Cette requête est similaire à la requête 3, mais elle filtre les joueurs qui ont passé plus de 2 commandes en utilisant la clause **HAVING**.

## Requête 5

***Trouver le pseudo du joueur et le nom de l'arme pour chaque commande.***

**SELECT t\_commande.fkJoueur AS IdJoueur, t\_joueur.jouPseudo, t\_detail\_commande.fkArme, t\_arme.armNom**

**FROM t\_commande**

**JOIN t\_joueur ON t\_commande.fkJoueur = t\_joueur.idJoueur**

**JOIN t\_detail\_commande ON t\_commande.idCommande = t\_detail\_commande.fkCommande**

**JOIN t\_arme ON t\_detail\_commande.fkArme = t\_arme.idArme;**

Cette requête relie les tables **t\_commande, t\_joueur**, **t\_detail\_commande**, et **t\_arme** pour afficher le pseudo du joueur et le nom de l'arme pour chaque commande passée.

## Requête 6

**Trouver le total dépensé par chaque joueur en ordonnant par le montant le plus élevé en premier, et limiter aux 10 premiers joueurs.**

**La 1ère colonne doit avoir pour nom "IdJoueur" et la 2ème colonne "TotalDepense"**

**SELECT fkJoueur AS IdJoueur, SUM(t\_arme.armPRIX) AS TotalDepense**

**FROM t\_commande**

**JOIN t\_detail\_commande ON t\_detail\_commande.fkCommande = t\_commande.idCommande**

**JOIN t\_arme ON t\_detail\_commande.fkArme = t\_arme.idArme**

**GROUP BY fkJoueur**

**ORDER BY TotalDepense DESC LIMIT 10;**

Cette requête calcule le montant total dépensé par chaque joueur en reliant les tables appropriées. Les résultats sont triés par le montant le plus élevé en premier, et seuls les 10 premiers joueurs sont inclus.

## Requête 7

***Récupérez tous les joueurs et leurs commandes, même s'ils n'ont pas passé de commande.***

***Dans cet exemple, même si un joueur n'a jamais passé de commande, il sera quand même listé, avec des valeurs `NULL` pour les champs de la table `t\_commande`.***

**SELECT t\_joueur.idJoueur, t\_joueur.jouPseudo, t\_commande.idCommande, t\_commande.comDate, t\_commande.comNumeroCommande**

**FROM t\_joueur**

**LEFT JOIN t\_commande ON t\_joueur.idJoueur = t\_commande.fkJoueur;**

Cette requête utilise une jointure de gauche (LEFT JOIN) pour récupérer tous les joueurs, qu'ils aient passé des commandes ou non. Les colonnes de la table **t\_commande** sont incluses, et si un joueur n'a pas passé de commande, les valeurs seront **NULL**.

## Requête 8

***Récupérer toutes les commandes et afficher le pseudo du joueur s’il existe, sinon afficher `NULL` pour le pseudo.***

**SELECT t\_commande.idCommande, t\_commande.comDate, t\_commande.comNumeroCommande, t\_joueur.jouPseudo**

**FROM t\_commande**

**RIGHT JOIN t\_joueur ON t\_commande.fkJoueur = t\_joueur.idJoueur;**

Cette requête utilise une jointure de droite (RIGHT JOIN) pour récupérer toutes les commandes, et elle affiche le pseudo du joueur s'il existe, sinon, le champ du pseudo sera **NULL**.

## Requête 9

***Trouver le nombre total d'armes achetées par chaque joueur (même si ce joueur n'a acheté aucune Arme).***

**SELECT fkJoueur, COUNT(idArme)**

**FROM t\_commande**

**LEFT JOIN t\_detail\_commande**

**ON t\_commande.idCommande = t\_detail\_commande.fkCommande**

**LEFT JOIN t\_arme**

**ON t\_detail\_commande.fkArme = t\_arme.idArme**

**GROUP BY fkJoueur;**

Cette requête trouve le nombre total d'armes achetées par chaque joueur, même s'ils n'ont acheté aucune arme. Elle utilise une jointure de gauche pour inclure tous les joueurs.

## Requête 10

***Trouver les joueurs qui ont acheté plus de 3 types d'armes différentes***

**SELECT t\_joueur.idJoueur, t\_joueur.jouPseudo**

**FROM t\_joueur**

**LEFT JOIN t\_commande ON t\_joueur.idJoueur = t\_commande.fkJoueur**

**LEFT JOIN t\_detail\_commande ON t\_commande.idCommande = t\_detail\_commande.fkCommande**

**GROUP BY t\_joueur.idJoueur, t\_joueur.jouPseudo**

**HAVING COUNT(DISTINCT t\_detail\_commande.fkArme) > 3;**

Cette requête identifie les joueurs qui ont acheté plus de 3 types d'armes différentes en utilisant une jointure de gauche et en appliquant la clause **HAVING** pour filtrer les résultats.

# Index

## Pourquoi certains index existent déjà

Toutes les « **id** » et les « **Foreign Key (FK)** » sont des index déjà existants sur la base de données.

## Avantages et désavantages des Index

### Avantages

Les index octroient des performances de lecture en accélérant les requêtes de lecture, en réduisant le nombre de lignes que le moteur de base de données doit parcourir.

Ils optimisent aussi les requêtes en aidant la base de données à choisir des stratégies plus efficaces.

### Désavantages

Espace disque supplémentaire, les index occupent de l’espace disque supplémentaire qui pourrait être utilisé autrement.

Coût dans le temps, la création d’index peut prendre du temps en particulier si on travaille sur de grandes tables.

## Où ajouter des Index pertinents

Cela dépend énormément des recherches qui vont être effectuer sur la base de données. On pourrait rajouter un index sur « **armPrix** » de la table « **t\_arme** » si on utilise fréquemment des requêtes qui filtres ou trient les armes par prix.

# Backup / Restore

## Backup

Pour effectuer un backup complet de la base de données **db\_space\_invaders**, il faut entrer la commande suivante :

**mysqldump -uroot -proot --databases db\_space\_invaders > backupDb.sql**

## Restore

Pour restore la base de données **db\_space\_invaders** depuis le fichier créé précédemment, il faut entrer la commande suivante :

**Mysql -uroot -proot < backupDb.sql**

# Conclusion

J’ai trouvé ce module très intéressant malgrès le fait que ça soit un module qui est plus tourné vers l’infrastructure, le SQL est un language qui pourra m’avérer très utile dans mes futures années dans l’informatique.