

**Rapport P-DB**

**106**



*Albion Bllaca*

*ETML - CIN2A*

*DGEP-ETAT DE VAUD*

Table des matières

[1 Introduction 3](#_Toc185429134)

[1.1 Description 3](#_Toc185429135)

[2 Modèle conceptuel de données 3](#_Toc185429136)

[2.1 Explications des tables 3](#_Toc185429137)

[2.2 Description des tables intermédiaire 4](#_Toc185429138)

[3 Exécution des script SQL 4](#_Toc185429139)

[3.1 Création de la base de données 4](#_Toc185429140)

[3.2 Insertion des données 5](#_Toc185429141)

[3.3 Requête 5](#_Toc185429142)

[4 Création des utilisateurs 10](#_Toc185429143)

[4.1 Création des rôles 10](#_Toc185429144)

[4.2 Attribution des privilèges aux rôles 10](#_Toc185429145)

[4.3 Création des utilisateurs et attribution des rôles 10](#_Toc185429146)

[4.4 Test des utilisateurs 10](#_Toc185429147)

[5 Index 11](#_Toc185429148)

[6 Backup / restore 11](#_Toc185429149)

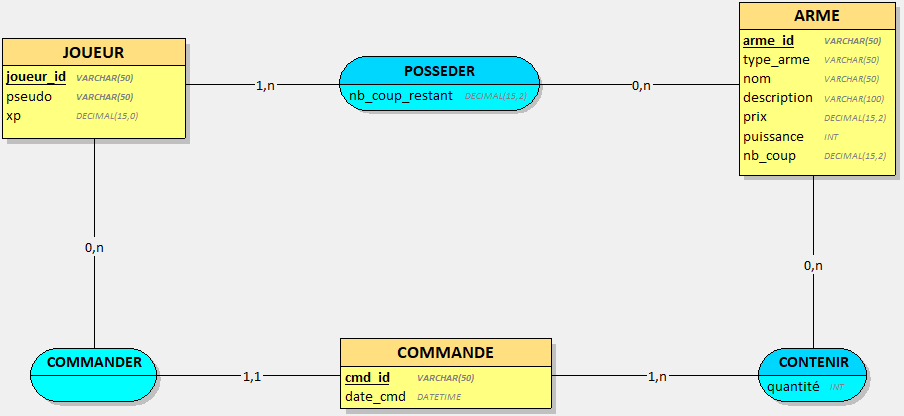
[7 Conclusion **Erreur ! Signet non défini.**](#_Toc185429150)

# Introduction

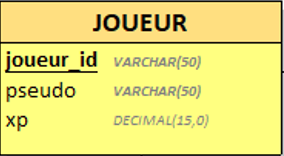
## Description

Le but du projet est de créer une base de données tout en suivant le cahier des charges avec ces fonctionnalités et contraintes. Le projet se base sur un jeux fictif appelé SpicyInvaders.

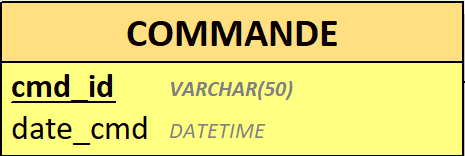
# Modèle conceptuel de données



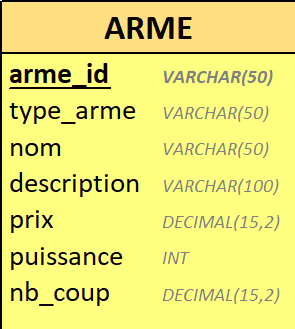
## Explications des tables



La table joueur permet de stocker l’ID du joueur, son pseudonyme et ses points d’expériences (XP)



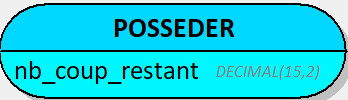
La table commande permet de mettre une identification a une commande et de stocker une date grâce à ces liaisons entre la table joueur et arme.

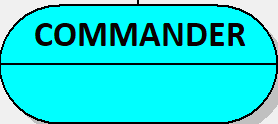


Cette table permet de stocker les armes et leur attribut comme le type, le nom, une description, un prix, une puissance de feu et le nombre d’utilisation restante dans l’arme, en partant du principe que l’arme ne peut plus être utilisée après le nombre de coup tiré

## Description des tables intermédiaire

La table posséder permet de faire le lien entre la table arme et la table joueur. Elle permet aussi de déterminer le nombre de coup restant au joueur pour ses armes





La table commander permet de faire le lien entre une commande contenant id, arme et quantité et un joueur en incluant l’id du joueur dans la table commande (en tant que clé étrangère).

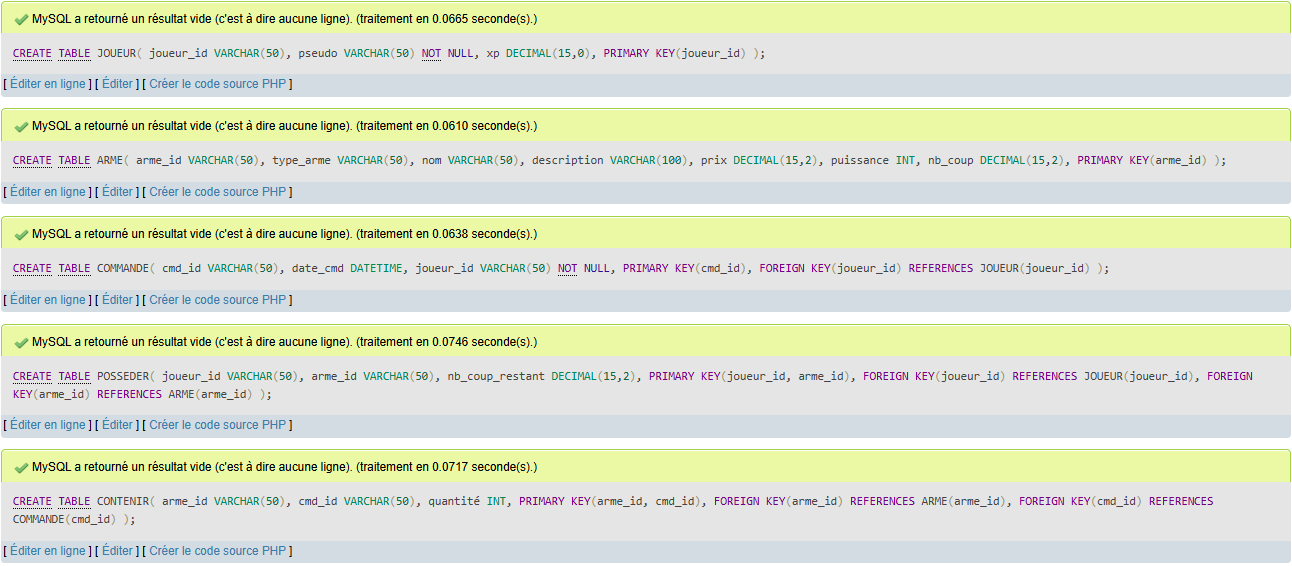
La table contenir permet de stocker le détail de chaque commande. Elle fait le lien entre la table **Commandes** et la table **Armes**, associant ainsi une arme commandée à un identifiant de commande, une date de commande, le joueur qui a passé la commande et la quantité commandée.



# Exécution des script SQL

## Création de la base de données

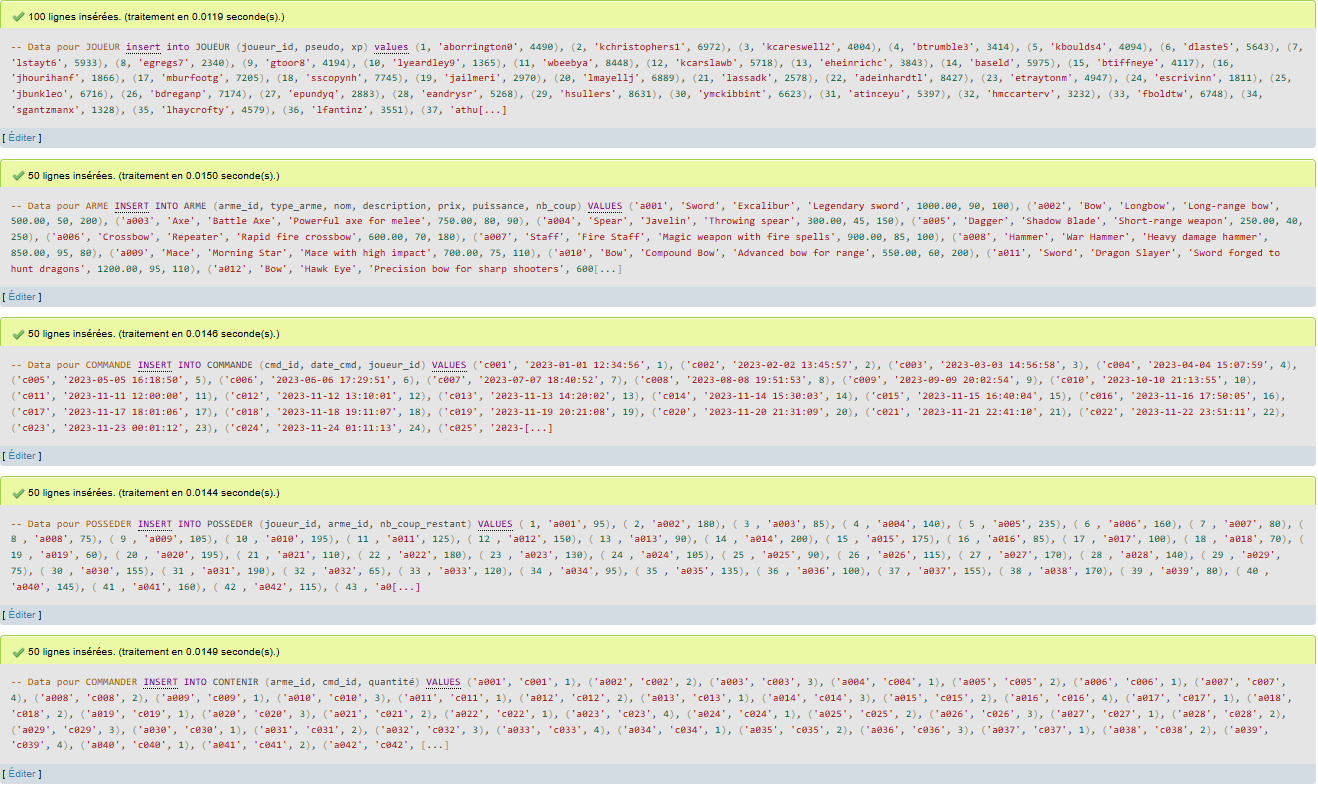
Le script SQL utilisé pour créer la base de données se nomme [SpicyInvaders.sql](../SQL/SpicyInvaders.sql).



*Le script s’exécute correctement sans erreurs.*

## Insertion des données

Le script SQL utilisé pour créer la base de données se nomme [data\_SpicyInvaders.sql](../SQL/data_SpicyInvaders.sql).

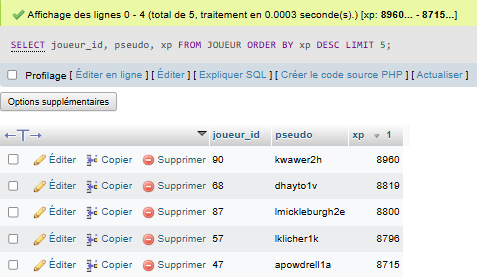


*Les données ont été insérée sans erreurs dans la base de données.*

## Requête

Les requêtes utilisées sont dans le fichier nommé [requetes.sql](../SQL/requetes.sql).

**Requête N°1** : **Sélectionner les 5 joueurs avec le meilleur score, classés dans l'ordre décroissant**



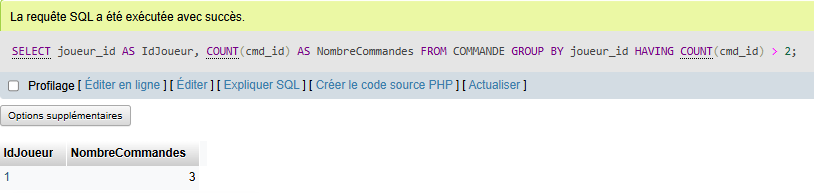
**Requête N°2 : Trouver le prix maximum, minimum et moyen des armes**



**Requête N°3 : Nombre total de commandes par joueur, trié par ordre décroissant**



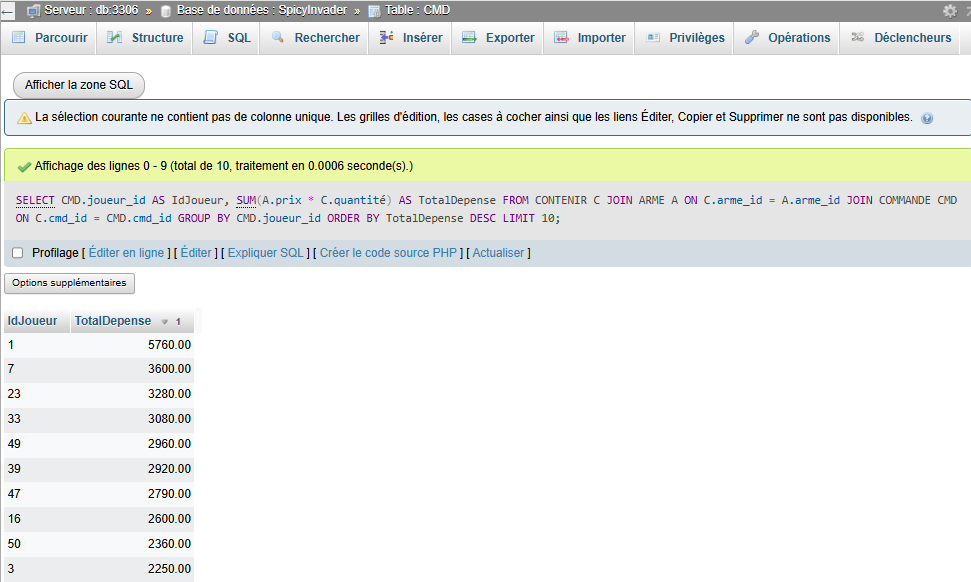
**Requête N°4 : Trouver les joueurs ayant passé plus de 2 commandes**



**Requête N°5 : Trouver le pseudo du joueur et le nom de l'arme pour chaque commande**



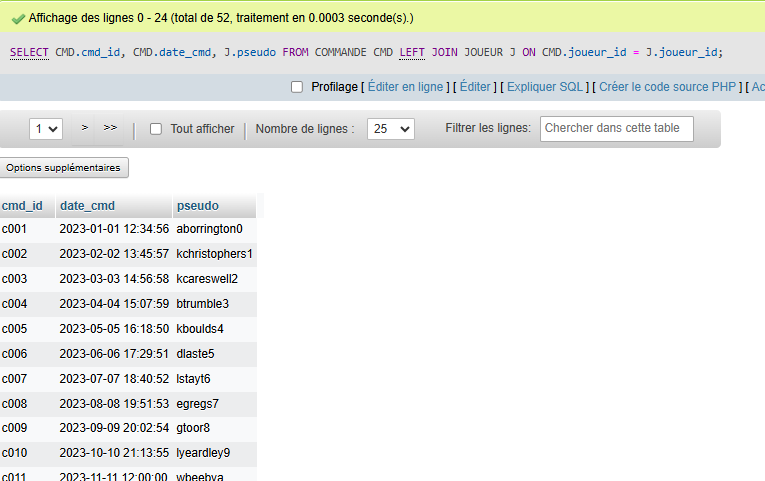
**Requête N°6 : Total dépensé par chaque joueur, ordonné par montant décroissant, limité aux 10 premiers joueurs**



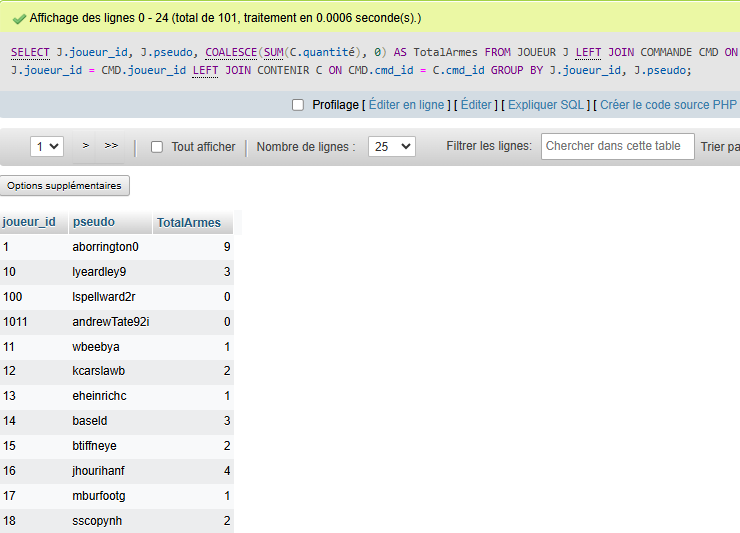
**Requête N°7 : Récupérer tous les joueurs et leurs commandes, même s'ils n'ont pas passé de commande**

****

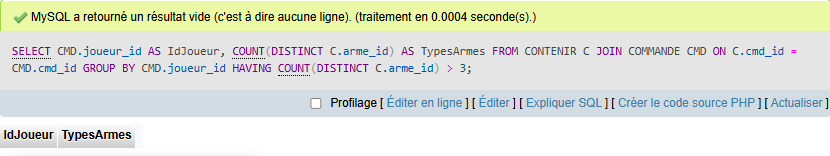
**Requête N°8 : Récupérer toutes les commandes et afficher le pseudo du joueur ou NULL si le joueur n'existe pas**

****

**Requête N°9 : Nombre total d'armes achetées par chaque joueur, y compris ceux qui n'ont acheté aucune arme**

****

**Requête N°10 : Trouver les joueurs ayant acheté plus de 3 types d'armes différentes**

****

# Création des utilisateurs

Tout d’abord nous commençons par créer des rôles. Ces rôles permettent d’avoir des droits assignés à des zones spécifiques, comme le droit à la lecture (SELECT), insertion (INSERT), suppression (DELETE), etc… Ces rôles seront par la suite attribués à des utilisateurs qui sont créer par la suite.

## Création des rôles

CREATE ROLE 'Role\_Administrateur';

CREATE ROLE 'Role\_Joueur';

CREATE ROLE 'Role\_Gestionnaire';

## Attribution des privilèges aux rôles

Attribution des privilèges Administrateur

GRANT ALL PRIVILEGES ON \* TO 'Role\_Administrateur';

Attribution des privilèges Joueur

GRANT SELECT ON ARME TO 'Role\_Joueur';

GRANT INSERT ON COMMANDE TO 'Role\_Joueur';

GRANT SELECT ON COMMANDE TO 'Role\_Joueur';

Attribution des privilèges au Gestionnaire de la boutique

GRANT SELECT ON JOUEUR TO 'Role\_Gestionnaire';

GRANT SELECT, UPDATE, DELETE ON ARME TO 'Role\_Gestionnaire';

GRANT SELECT ON COMMANDE TO 'Role\_Gestionnaire';

## Création des utilisateurs et attribution des rôles

CREATE USER 'Administrateur'@'localhost' IDENTIFIED BY 'root';

GRANT 'Role\_Administrateur' TO 'Administrateur'@'localhost';

SET DEFAULT ROLE 'Role\_Administrateur' TO 'Administrateur';

CREATE USER 'Joueur'@'localhost' IDENTIFIED BY 'root';

GRANT 'Role\_Joueur' TO 'Joueur'@'localhost';

SET DEFAULT ROLE 'Role\_Joueur' TO 'Joueur';

CREATE USER 'Gestionnaire\_boutique'@'localhost' IDENTIFIED BY 'root';

GRANT 'Role\_Gestionnaire' TO 'Gestionnaire\_boutique'@'localhost';

SET DEFAULT ROLE 'Role\_Gestionnaire' TO 'Gestionnaire\_boutique';

## Test des utilisateurs

SHOW GRANTS FOR 'Administrateur'@'localhost';

SHOW GRANTS FOR 'Joueur'@'localhost';

SHOW GRANTS FOR 'Gestionnaire\_boutique'@'localhost';

# Index

Créer un index sur les points d'expérience (xp) des joueurs dans une table JOUEUR permet d'améliorer les performances des requêtes qui filtrent ou trient les données en fonction de cette colonne.

CREATE INDEX point\_exp ON JOUEUR (xp);

# Backup / restore

Voici la commande permettant de faire un backup (sauvegarde)

mysqldump -u root -proot SpicyInvader > SpicyInvader\_back.sql

La première commande utilisée est mysqldump, car c’est une application dédiée à effectuer les sauvegardes de bases de données. Ensuite, on spécifie le nom d’utilisateur et le mot de passe avec les options -u root et -proot. Après cela, il faut indiquer la base de données à sauvegarder, dans ce cas, "SpicyInvaders". Enfin, le symbole > est utilisé pour rediriger la sortie et spécifier le nom du fichier dans lequel on souhaite enregistrer la base de données.

Le fichier de backup est disponible sous le nom de [SpicyInvader\_back.sql](../SQL/SpicyInvader_back.sql).

Pour l’export c’est presque pareil, la différence c’est que cette fois-ci on utilise mysql comme application et on inverse le jalon. Bien sur comme cette fois-ci nous restaurons une base il ne faut pas spécifier de base de données.

Voici la commande permettant de faire un restore (restauration)

mysql -u root -proot < SpicyInvader\_back.sql