Projet DB 106



**Rapport de Projet**

Zeqiri Amir – CIN2B

ETML, Vennes – A01

32 Périodes

Maître : M. Charmier

Table des matières

[1 Description du projet dans son ensemble 3](#_Toc181718578)

[1.1 Sujet 3](#_Toc181718579)

[1.2 Spécificités DB 3](#_Toc181718580)

[2 Pratique 4](#_Toc181718581)

[2.1 Réaliser le MCD 4](#_Toc181718582)

[3 Conclusion 4](#_Toc181718583)

# Description du projet dans son ensemble

## Sujet

Travailler avec une base de données sur la thématique « SpaceInvader »

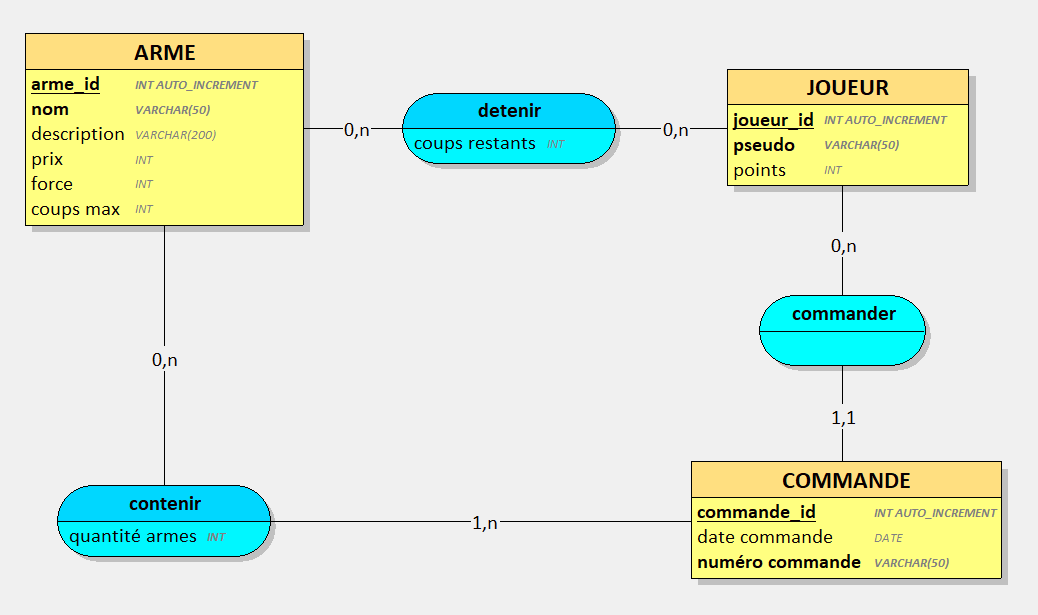
## Spécificités DB

Au niveau de la DB, il nous est demandé de :

* Maitriser la réalisation de MCD avec le logiciel Looping
* Maitriser la création de base de données et les importations des données dans MySQL
* Maitriser la gestion des utilisateurs en créant des rôles, des utilisateurs, à accorder des droits, etc.
* Maitriser les requêtes de sélection en répondant à 10 requêtes différentes et les expliquer en détail.
* Maitriser les index en répondant aux trois questions du cahier des charges
* Maitriser les Backups / Restores en effectuant une sauvegarde et restaurant une sauvegarde et expliquer en détail

# Pratique

## Réaliser le MCD



## Création de la base de données et importation des données

### Création de la DB

Voici le script que j’ai récupéré sur Looping qui m’a servi à créer la base de données :

CREATE DATABASE db\_space\_invaders;

USE db\_space\_invaders;

CREATE TABLE t\_arme(

arme\_id INT AUTO\_INCREMENT,

nom VARCHAR(50),

description VARCHAR(200),

prix INT,

force INT,

coups\_max INT,

PRIMARY KEY(arme\_id),

UNIQUE(nom)

);

CREATE TABLE t\_joueur(

joueur\_id INT AUTO\_INCREMENT,

pseudo VARCHAR(50),

points INT,

PRIMARY KEY(joueur\_id),

UNIQUE(pseudo)

);

CREATE TABLE t\_commande(

commande\_id INT AUTO\_INCREMENT,

date\_commande DATE,

numero\_commande VARCHAR(50),

joueur\_fk INT NOT NULL,

PRIMARY KEY(commande\_id),

UNIQUE(numero\_commande),

FOREIGN KEY(joueur\_fk) REFERENCES t\_joueur(joueur\_id)

);

CREATE TABLE t\_contenir(

arme\_fk INT,

commande\_fk INT,

quantite\_armes INT NOT NULL,

PRIMARY KEY(arme\_fk, commande\_fk),

FOREIGN KEY(arme\_fk) REFERENCES t\_arme(arme\_id),

FOREIGN KEY(commande\_fk) REFERENCES t\_commande(commande\_id)

);

CREATE TABLE t\_detenir(

arme\_fk INT,

joueur\_fk INT,

coups\_restants INT,

PRIMARY KEY(arme\_fk, joueur\_fk),

FOREIGN KEY(arme\_fk) REFERENCES t\_arme(arme\_id),

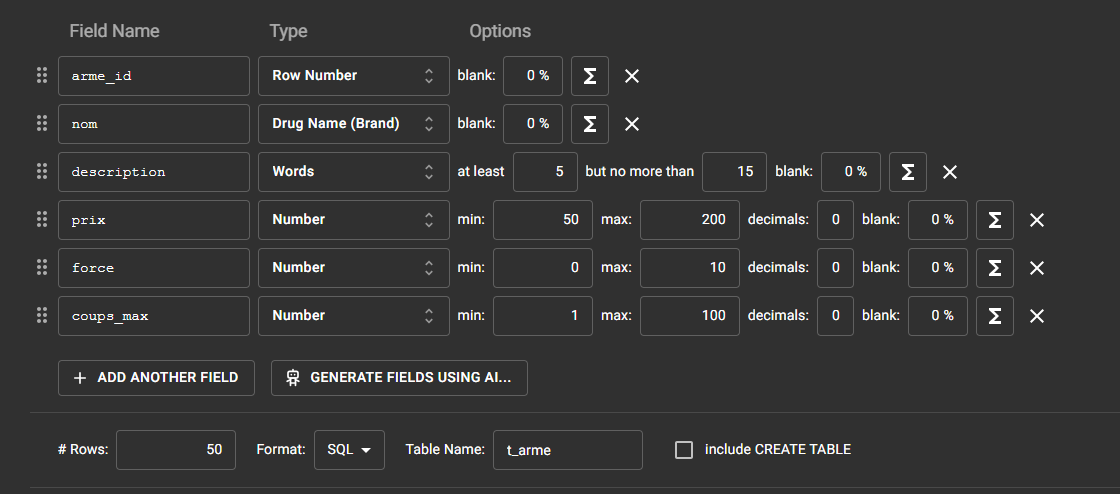
FOREIGN KEY(joueur\_fk) REFERENCES t\_joueur(joueur\_id)

);

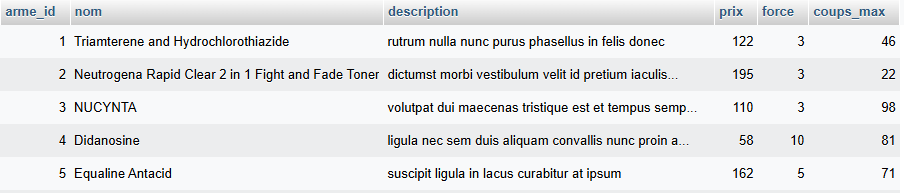
### Création et importation des données

Pour créer 50 enregistrements par table, j’ai utilisé [Mockaroo](https://www.mockaroo.com/). Avec ce site j’ai généré des données cohérentes avec les noms des champs.

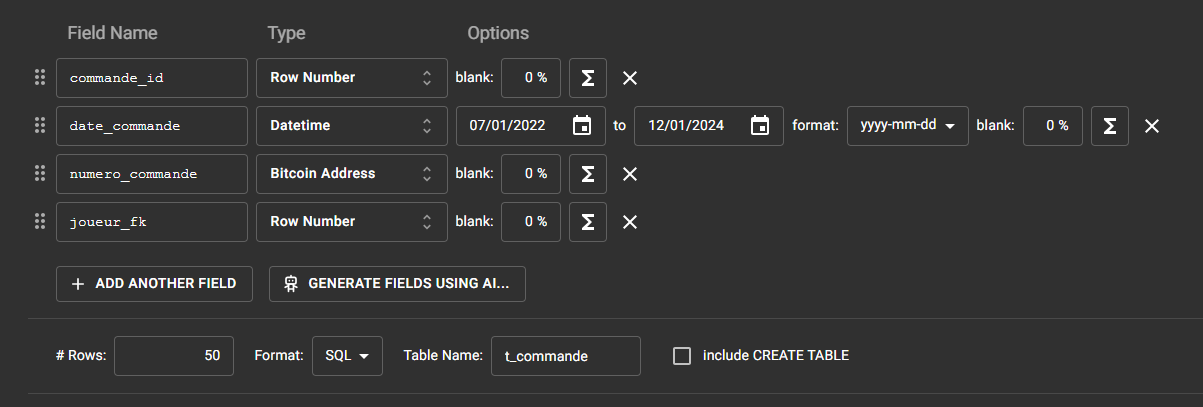
Pour la table ‘**t\_arme**’, voici les types de données que je lui ai demandé :



Et voici les 5 premiers enregistrements de la table :



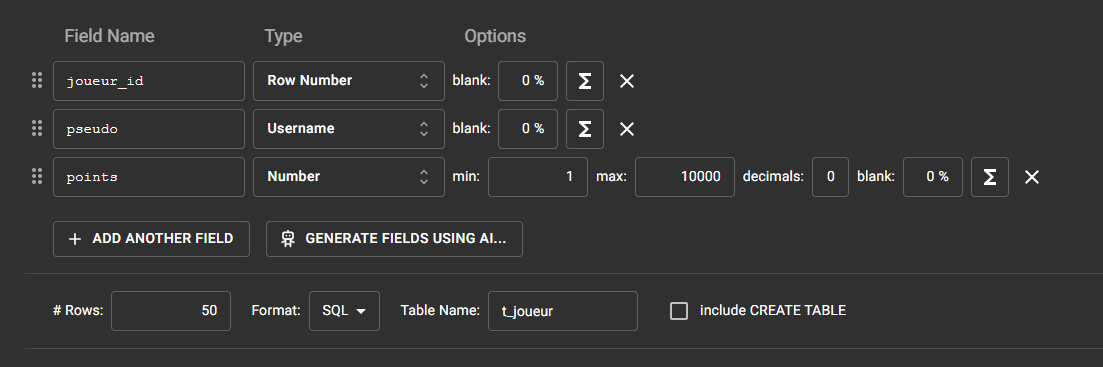
Pour la table ‘**t\_commande**’, voici les types de données que je lui ai demandé :



Et voici les 5 premiers enregistrements de la table :



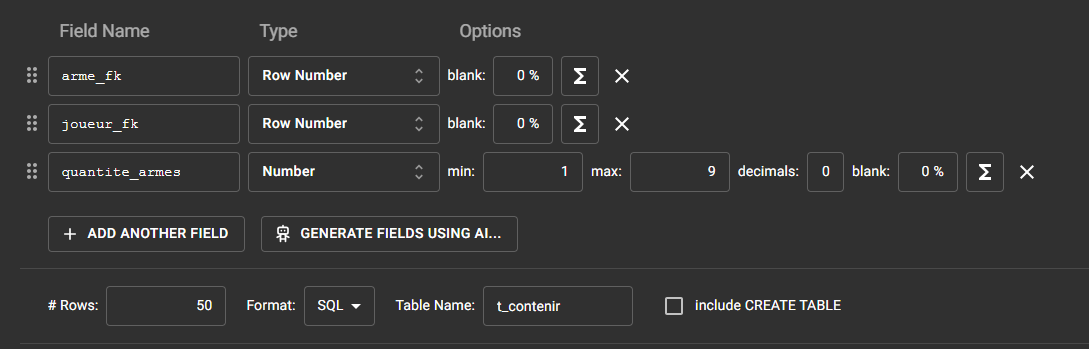
Pour la table ‘**t\_joueur**’, voici les types de données que je lui ai demandé :



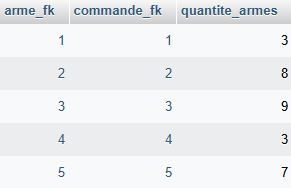
Et voici les 5 premiers enregistrements de la table :



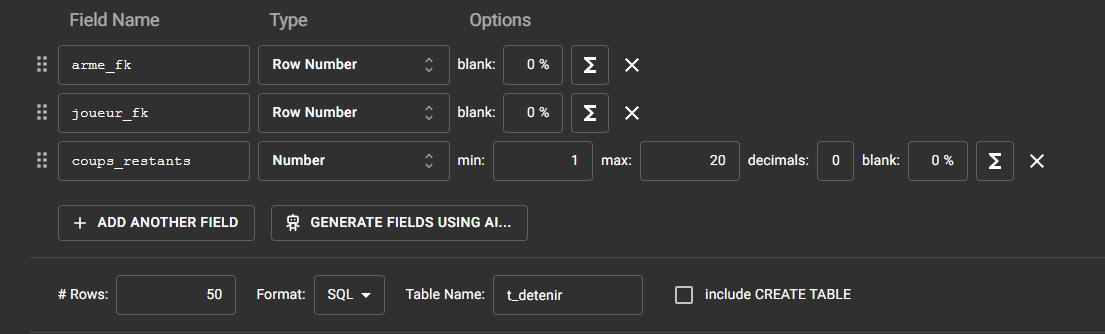
Pour la table ‘**t\_contenir**’, voici les types de données que je lui ai demandé :



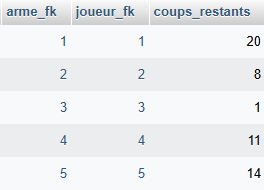
Et voici les 5 premiers enregistrements de la table :



Pour la table ‘**t\_detenir**’, voici les types de données que je lui ai demandé :



Et voici les 5 premiers enregistrements de la table :



Pour importer les données dans MySQL, j’ai sélectionné la table désirée et j’ai copier-coller le fichier sql qui comporte les 50 INSERT INTO dans l’espace SQL de PhpMyAdmin.

## Gestions des utilisateurs

Il nous est demandé de créer trois utilisateurs, l’un se nommera "Administrateur", l’autre "Joueur" et le dernier "Gestionnaire". L’administrateur du jeu doit pouvoir créer, lire, mettre à jour et supprimer toutes les tables, il pourra également gérer les différents utilisateurs et leurs privilèges.

Le joueur doit pouvoir lire les informations des armes (voir quelles armes il peut acheter), créer une commande et lire toutes les commandes.

Le gestionnaire de la boutique doit pouvoir 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, modifications des prix, etc.) et lire toutes les commandes

Voici comment j’ai procédé :

Pour commencer, j’ai créé les trois rôles nécessaires. Dans le cahier des charges il est écrit ceci : "Pour chaque « catégorie d’utilisateurs », on pourrait être amener à créer plusieurs voire un grand nombre d’utilisateurs." C’est pour cela qu’il est préférable de créer des rôles et non directement des utilisateurs.

Voici comment j’ai procédé :

CREATE ROLE 'Administrateur';

CREATE ROLE 'Joueur';

CREATE ROLE 'Gestionnaire';

Ces trois commandes ont créé les trois rôles nécessaires.

Maintenant je vais attribuer les différents privilèges selon le cahier des charges au rôle de l’administrateur.

Voici comment j’ai procédé :

GRANT ALL PRIVILEGES ON db\_space\_invaders.\* TO 'Administrateur' WITH GRANT OPTION;

Avec cette commande, j’accorde tous les privilèges sur toutes les tables de la base de données db\_space\_invaders au rôle Administrateur. Les utilisateurs de ce rôle pourront à leur tour accorder ces mêmes privilèges à la même table et base de données à d’autres rôles ou utilisateurs.

Ensuite, je vais attribuer les différents privilèges selon le cahier des charges au rôle du joueur.

Voici comment j’ai procédé :

GRANT SELECT ON db\_space\_invaders.t\_arme TO 'Joueur';

GRANT SELECT ON db\_space\_invaders.t\_commande TO 'Joueur';

GRANT INSERT ON db\_space\_invaders.t\_commande TO 'Joueur';

Avec ces trois lignes de commande, j’attribue les privilèges Select (Lecture) sur la table t\_arme et t\_commande de la base de données db\_space\_invaders au rôle Joueur. Également, j’attribue les privilèges Insert (Ajout de données) sur la table t\_commande de la base de données db\_space\_invaders au rôle Joueur.

Pour en finir avec les rôles, je vais attribuer les différents privilèges selon le cahier des charges au rôle du gestionnaire de la boutique.

Voici comment j’ai procédé :

GRANT SELECT ON db\_space\_invaders.t\_joueur TO 'Gestionnaire';

GRANT SELECT ON db\_space\_invaders.t\_arme TO 'Gestionnaire';

GRANT SELECT ON db\_space\_invaders.t\_commande TO 'Gestionnaire';

GRANT INSERT ON db\_space\_invaders.t\_arme TO 'Gestionnaire';

GRANT DELETE ON db\_space\_invaders.t\_arme TO 'Gestionnaire';

GRANT UPDATE ON db\_space\_invaders.t\_arme TO 'Gestionnaire';

Avec ces six lignes de commande, j’attribue les privilèges Select (Lecture) sur la table t\_joueur, t\_arme et t\_commande de la base de données db\_space\_invaders au rôle Gestionnaire. Également, j’attribue les privilèges Insert (Ajout de données), Delete (Supression) et Update (Mise à jour) sur la table t\_arme de la base de données db\_space\_invaders au rôle Gestionnaire.

Je vais maintenant, créer les trois utilisateurs différents.

Voici comment j’ai procédé :

CREATE USER 'Admin'@'localhost' IDENTIFIED BY 'admin12345';

CREATE USER 'Jojo'@'localhost' IDENTIFIED BY 'Jojo32';

CREATE USER 'Gest'@'localhost' IDENTIFIED BY 'gestt.200';

Ces trois lignes de commande m’ont permis de créer les trois utilisateurs. Je leurs ai défini un mot de passe chacun.

Je vais, pour finir, attribuer les rôles que je viens de créer aux utilisateurs qui leur est attribué.

Voici comment j’ai procédé :

GRANT 'Administrateur' TO 'Admin'@'localhost';

GRANT 'Joueur' TO 'Jojo'@'localhost';

GRANT 'Administrateur' TO 'Gest'@'localhost';

Avec ces trois lignes, je viens d’attribuer les rôles que j’ai créé précédemment aux utilisateurs que je viens de créer

J’ai, maintenant, fini la gestion des utilisateurs. Tous les privilèges, rôles et utilisateurs sont opérationnels et comme demandé dans le cahier des charges.

## Requêtes de sélection

### Requête 1

SELECT \*

FROM t\_joueur

ORDER BY t\_joueur.points

DESC

LIMIT 5;

On nous demande les 5 joueurs qui ont le meilleur score. Je sais que je vais devoir un LIMIT de 5, je sais que je dois sélectionner tout des joueurs. Je vais donc utiliser un ORDER BY sur les points des joueurs sans oublier le DESC parce que sinon ça affiche du plus nul au plus fort.

### Requête 2

SELECT MAX(t\_arme.prix) AS PrixMaximum,

MIN(t\_arme.prix) AS PrixMinimum,

AVG(t\_arme.prix) AS PrixMoyen

FROM t\_arme;

On nous demande le prix maximum, minimum, moyen d’une arme. Je sais donc que je dois utiliser MAX, MIN, AVG sur le prix d’une arme. Il nous est demandé de renommer les colonnes, donc j’utilise des AS pour chaque champ.

### Requête 3

SELECT t\_commande.joueur\_fk AS idJoueur,

COUNT(t\_commande.numero\_commande) AS NombreCommandes

FROM t\_commande

GROUP BY t\_commande.joueur\_fk

ORDER BY COUNT(t\_commande.numero\_commande) DESC;

On nous demande de trouver le nombre total de commandes par joueur et les trier du plus grand au plus petit. Comme c’est demandé **par joueur**, alors je vais utiliser un GROUP BY sur le joueur. Je sélectionne l’id du joueur dans la table commande, je vais utiliser un COUNT sur le numéro de commande pour compter le nombre de commande. Je n’utilise pas le DISTINCT car chaque numéro de commande est unique. J’utilise le ORDER BY pour ordonner le résultat sans oublier le DESC pour les ordonner du plus grand au plus petit. On nous demande de renommer les colonnes, pour cela, j’utilise des AS.

### Requête 4

SELECT t\_commande.joueur\_fk AS idJoueur,

COUNT(t\_commande.numero\_commande) AS NombresCommandes

FROM t\_commande

GROUP BY t\_commande.joueur\_fk

HAVING COUNT(t\_commande.numero\_commande) > 2 ;

On nous demande de trouver les joueurs qui ont plus de 2 commandes. Je sais donc que je vais devoir utiliser un COUNT sur les commandes dans la table des commandes. Je sélectionne aussi les joueurs en eux-mêmes de la table commandes. Le résultat doit être regroupé par chaque joueur car un joueur peut avoir plusieurs commandes et donc le GROUP BY est nécessaire, il va "distinct" le résultat. Comme il nous est demandé de trouver uniquement les joueurs à plus de 2 commandes, alors je vais utiliser un HAVING sur le COUNT du nombre de commandes, que je dois mettre à plus de 2.

### Requête 5

SELECT t\_joueur.pseudo, t\_arme.nom, t\_commande.numero\_commande

FROM t\_joueur

JOIN t\_commande ON t\_commande.joueur\_fk = t\_joueur.joueur\_id

JOIN t\_contenir ON t\_commande.commande\_id = t\_contenir.commande\_fk

JOIN t\_arme ON t\_contenir.arme\_fk = t\_arme.arme\_id ;

On nous demande de trouver le pseudo et je nom de l’arme pour chaque commande. Comme il est précisé pour **chaque commande** on pourrait croire qu’il faudrait utiliser un GROUP BY, mais non, car chaque commande est unique, il est donc pas nécessaire. Je sélectionne les pseudos, les noms des armes et les numéros de commande, je vais donc joindre plusieurs tables entres elles : joueur, commande, contenir et arme. Il est nécessaire de joindre ces différentes tables car j’utilise des champs de ces tables et je dois joindre la table contenir car je dois lier les tables commande et arme entres elles.

### Requête 6

SELECT t\_joueur.pseudo AS idJoueur,

SUM(t\_arme.prix) AS TotalDepense

FROM t\_joueur

JOIN t\_commande ON t\_joueur.joueur\_id = t\_commande.joueur\_fk

JOIN t\_contenir ON t\_commande.commande\_id = t\_contenir.commande\_fk

JOIN t\_arme ON t\_contenir.arme\_fk = t\_arme.arme\_id

GROUP BY t\_joueur.pseudo

ORDER BY SUM(t\_arme.prix)

DESC

LIMIT 10 ;

On nous demande de trouver le total dépensé par chaque joueur en les ordonnant par le montant le plus élevé en premier et limiter les réponses à 10. Comme on nous demande un total, alors je vais devoir utiliser un SUM sur le prix de l’arme de la table

### Requête 7

SELECT t\_joueur.pseudo, t\_commande.numero\_commande

FROM t\_joueur

LEFT JOIN t\_commande ON t\_commande.joueur\_fk = t\_joueur.joueur\_id ;

### Requête 8

SELECT t\_commande.commande\_id, t\_joueur.pseudo

FROM t\_commande

LEFT JOIN t\_joueur ON t\_joueur.joueur\_id = t\_commande.joueur\_fk;

### Requête 9

SELECT t\_joueur.pseudo, SUM(t\_contenir.quantite\_armes) AS "Nombre d'armes achetés"

FROM t\_joueur

LEFT JOIN t\_commande ON t\_joueur.joueur\_id = t\_commande.joueur\_fk

JOIN t\_contenir ON t\_commande.commande\_id = t\_contenir.commande\_fk

GROUP BY t\_joueur.pseudo ;

### Requête 10

SELECT t\_joueur.pseudo

FROM t\_joueur

JOIN t\_commande ON t\_commande.joueur\_fk = t\_joueur.joueur\_id

JOIN t\_contenir ON t\_contenir.commande\_fk = t\_commande.commande\_id

GROUP BY t\_joueur.pseudo

HAVING COUNT(DISTINCT t\_contenir.arme\_fk) > 3 ;

## Création des index

### Pourquoi certains index existent déjà ?

MySQL créé des index automatiquement



### Quels sont les avantages et les inconvénients des index ?

* Améliorations des performances de recherche
* Optimisation des requêtes
* Ça prend de la place en mémoire
* Ça ralentit les requêtes d’insertion, modification et suppression

### Sur quel champ, cela pourrait être pertinent d’ajouter un index ?

Il serait pertinent d’ajouter un index simple dans la table ‘t\_arme’, pour le champ ‘force’.

Les index sont utiles lorsque l’on utilise des WHERE, ORDER BY, GROUP BY. On pourrait être amené à mettre dans l’ordre la force de l’arme, ou à sélectionner les armes avec une force minimum ou maximum.

Je pourrais même créer un index composite des champs : ‘force’, ‘prix’, ‘coups\_max’. Si on est amené à rechercher des armes en utilisant souvent ces critères. Cela améliorera les performances de recherche comme dit dans les avantages.

## Backup / Restore

### Backup

Pour faire un backup de la base de données ‘db\_space\_invaders’, il suffit d’entrer cette ligne de commande :'

docker exec -i db mysqldump -u root -proot --databases db\_space\_invaders > db\_space\_invaders.sql;

**Explications :**

docker exec : Cela permet d’exécuter une commande dans Docker

-i : Cela spécifie ‘interactive’. Il faut utiliser cette option lorsque qu’il y des données en jeu.

db : Cela spécifie le container Docker que l’on sélectionne

mysqldump : J’inscris cette commande pour effectuer des sauvegardes de bases de données MySQL.

-uroot : Cela spécifie le nom d'utilisateur MySQL, dans ce cas, "root".

-proot : Cela spécifie le mot de passe MySQL, dans ce cas, "root". Nous pouvons également uniquement écrire -p et indiquer le mot de passe de manière plus sécurisée.

--databases db\_space\_invaders : Cela spécifie la base de données que je souhaite sauvegarder. Dans mon cas, la base de données s'appelle "db\_space\_invaders".

> db\_space\_invaders.sql : Cela redirige la sortie de la commande vers un fichier nommé "db\_space\_invaders.sql". Toutes les données de la base de données seront enregistrées dans ce fichier.

### Restore

Pour faire un restore du bump (fichier de sauvegarde sql d’une base de données) crée ci-dessus, il suffit d’entrer cette ligne de commande :

docker exec -i db mysql -u root -proot < db\_space\_invaders.sql;

**Explications :**

docker exec : Cela permet d’exécuter une commande dans Docker

-i : Cela spécifie ‘interactive’. Il faut utiliser cette option lorsque qu’il y des données en jeu.

db : Cela spécifie le container Docker que l’on sélectionne

mysql : C'est la commande que vous voulez exécuter à l'intérieur du conteneur Docker.

-uroot : Cela spécifie le nom d'utilisateur MySQL, dans ce cas, "root".

-proot : Cela spécifie le mot de passe MySQL, dans ce cas, "root". Nous pouvons également uniquement écrire -p et indiquer le mot de passe de manière plus sécurisée.

< sp\_space\_invader.sql : Cela signifie que je prends le contenu du fichier nommé "sp\_space\_invader.sql" et l'utiliser comme entrée pour la commande MySQL.

# Conclusion