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

1.1 Sujet

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

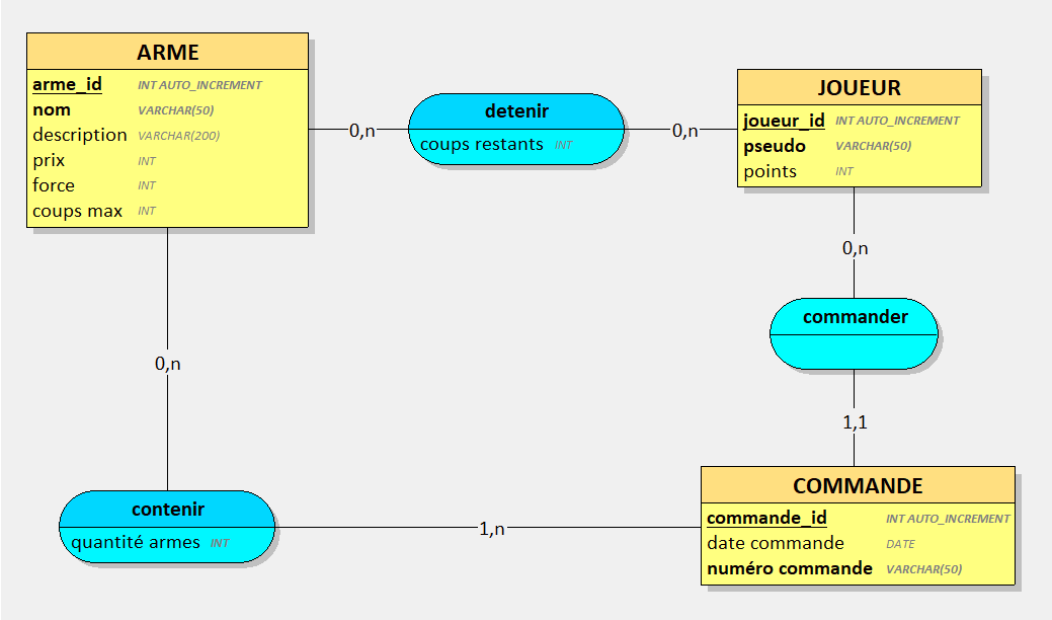
1.2 Spécificités DB

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

2 PRATIQUE

2.1 Réaliser le MCD



2.2 Création de la base de données et importation des données

2.2.1 Création de la DB

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

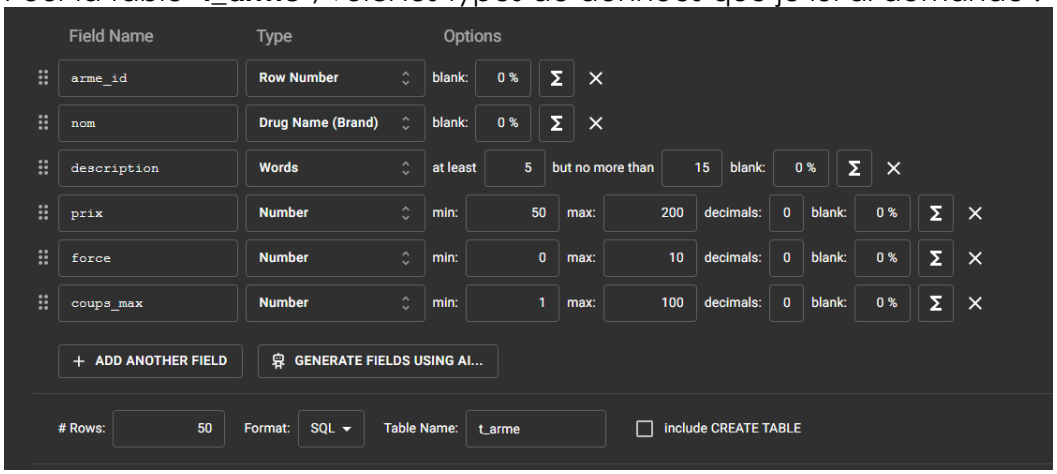
```
CREATE DATABASE db_space_invaders;
USE db_space_invaders;

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)
);
```

2.2.2 *Création et importation des données*

Pour créer 50 enregistrements par table, j'ai utilisé Mockaroo. Avec ce site j'ai généré des données cohérentes avec les noms des champs.

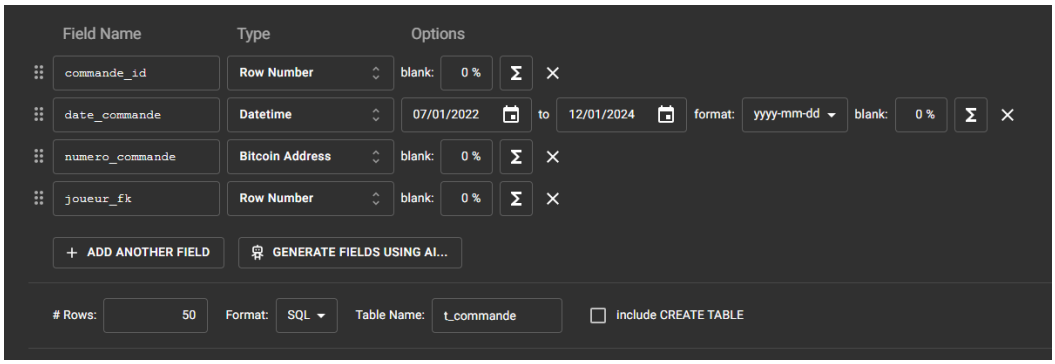
Pour la table '**t_arme**', voici les types de données que je lui ai demandé :



Et voici les 5 premiers enregistrements de la table :

arme_id	nom	description	prix	force	coups_max
1	Triamterene and Hydrochlorothiazide	rutrum nulla nunc purus phasellus in felis donec	122	3	46
2	Neutrogena Rapid Clear 2 in 1 Fight and Fade Toner	dictumst morbi vestibulum velit id pretium iaculis...	195	3	22
3	NUCYNTA	volutpat dui maecenas tristique est et tempus semp...	110	3	98
4	Didanosine	ligula nec sem duis aliquam convallis nunc proin a...	58	10	81
5	Equaline Antacid	suscipit ligula in lacus curabitur at ipsum	162	5	71

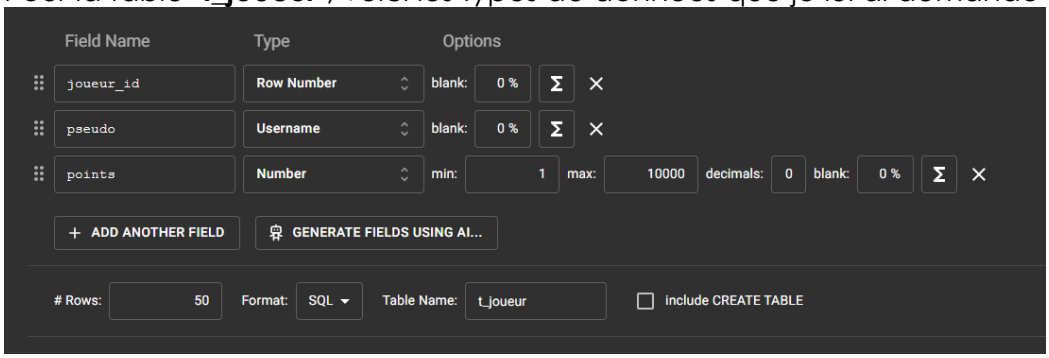
Pour la table '**t_commande**', voici les types de données que je lui ai demandé :



Et voici les 5 premiers enregistrements de la table :

commande_id	date_commande	numero_commande	joueur_fk
1	2023-04-25	124znjrEitv11gXXtVuo3QDeXmYWk2obSB	1
2	2024-07-08	1KYDWWYBydcb2jpW3N7NyhEagZxbbi56ff	2
3	2023-12-12	1HGDyszKGDuciZ64mbDUfZ9YPaNZMx45hw	3
4	2023-04-04	1H13wsd3JAp5k2xHPfmLzWF5jV1arxRG6u	4
5	2024-08-26	145Gdsbm75UzD5uFtuFuoaeoDrB9krPEzU	5

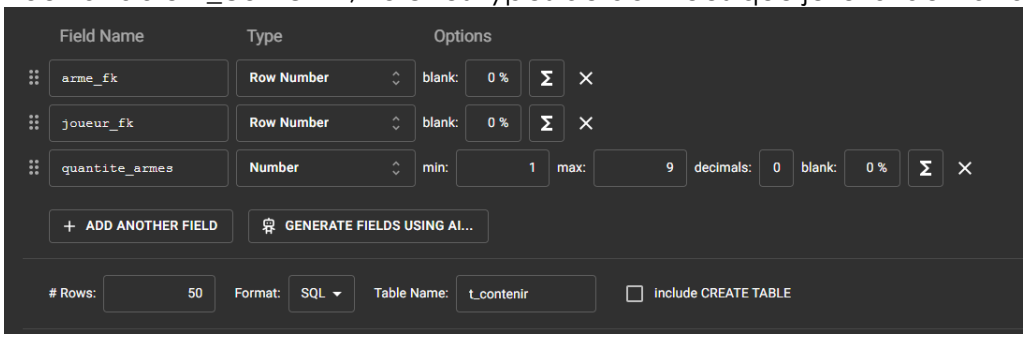
Pour la table '**t_joueur**', voici les types de données que je lui ai demandé :



Et voici les 5 premiers enregistrements de la table :

joueur_id	pseudo	points
1	jmorison0	5347
2	cmccool1	5345
3	visham2	1045
4	broo3	4910
5	lmunkley4	8609

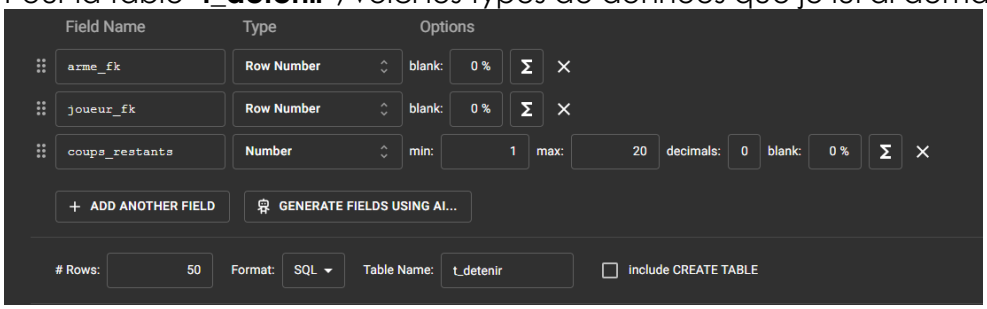
Pour la table '**t_contenir**', voici les types de données que je lui ai demandé :



Et voici les 5 premiers enregistrements de la table :

arme_fk	commande_fk	quantite_armes
1	1	3
2	2	8
3	3	9
4	4	3
5	5	7

Pour la table '**t_detenir**', voici les types de données que je lui ai demandé :



Et voici les 5 premiers enregistrements de la table :

arme_fk	joueur_fk	coups_restants
1	1	20
2	2	8
3	3	1
4	4	11
5	5	14

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.

2.3 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.

2.4 Requêtes de sélection

2.4.1 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.

2.4.2 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.

2.4.3 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.

2.4.4 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. On nous demande de renommer une colonne, pour cela, j'utilise un AS.

2.4.5 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 dois 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.

2.4.6 *Requête 6*

```
SELECT t_commande.joueur_fk AS idJoueur,
SUM(t_arme.prix * t_contenir.quantite_armes) AS TotalDepense
FROM t_commande
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_commande.joueur_fk
ORDER BY SUM(t_arme.prix * t_contenir.quantite_armes) 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 dois utiliser un SUM sur le prix de l'arme de la table multiplié par la quantité d'armes de la table commande, je sélectionne aussi le joueur depuis la table commande. Je dois joindre plusieurs tables entres elles : commande, contenir, arme. Il est nécessaire de joindre ces différentes tables car j'utilise des champs de ces tables. Comme c'est précisé par **chaque joueur** je dois utiliser un GROUP BY sur le joueur que j'ai sélectionné dans le SELECT. Il nous est demandé d'ordonner les résultats, j'utilise un ORDER BY sur la somme utilisée, sans oublier le DESC car on doit ordonner du montant le plus élevé au plus petit. Il nous est demandé de limiter les résultats aux 10 premiers joueurs, j'utilise donc un LIMIT à 10. On nous demande de renommer les colonnes, pour cela, j'utilise des AS.

2.4.7 *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 ;
```

On nous demande de trouver tous les joueurs et leurs commandes. Je sélectionne le pseudo du joueur et le numéro de commande. Tout ça de la table joueur que je joints à la table commande avec un LEFT JOIN car il est précisé que même un joueur qui n'a pas de commande sera affiché.

2.4.8 Requête 8

```
SELECT t_commande.numero_commande, t_joueur.pseudo
FROM t_commande
LEFT JOIN t_joueur ON t_joueur.joueur_id = t_commande.joueur_fk;
```

On nous demande de trouver toutes les commandes et le pseudo du joueur s'il existe. Je sélectionne le numéro de commande et le pseudo du joueur. Tout ça de la table commande que je joints à la table joueur avec un LEFT JOIN car il est précisé d'afficher 'NULL' pour le pseudo si la commande n'a pas de joueur.

2.4.9 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 ;
```

On nous demande de trouver le nombre total d'armes achetées par chaque joueur même si ce joueur n'a acheté aucune arme. Comme c'est précisé par **chaque joueur** je dois utiliser un GROUP BY sur le joueur que j'ai sélectionné dans le SELECT. J'effectue la somme des quantités d'armes. Tout ça de la table joueur que je joints à la table commande avec un LEFT JOIN car il est précisé d'afficher le joueur même s'il n'a acheté aucune arme. De plus je joints la table contenir, car je m'en sers pour la somme de la quantité d'arme. J'ajoute un AS sur la somme pour rendre le champ plus compréhensif, même si cela n'est pas demandé.

2.4.10 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 ;
```

On nous demande de trouver les joueurs qui ont acheté plus de 3 types d'armes différentes. Je sélectionne donc le pseudo du joueur. Je dois joindre plusieurs tables entres elles : joueur, commande et contenir. 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. Je dois utiliser un GROUP BY car j'utilise un HAVING COUNT (DISTINCT). J'utilise cet HAVING COUNT (DISTINCT) pour compter les armes de manière distincte, comme il est précisé le type d'arme et non l'arme et je le mets à partir de 3 comme demandé.

2.5 Création des index

2.5.1 Pourquoi certains index existent déjà ?

MySQL créé des index automatiquement.

Pour les clés primaires, il crée un index pour garantir l'unicité et accélérer les recherches. Pour les clés étrangères, il ajoute un index pour faciliter les jointures. Pour les clés uniques, il impose l'unicité et optimise les requêtes.

Printsceen tiré du module 106 :

MYSQL CRÉÉ CERTAINS INDEX AUTOMATIQUEMENT

- pour une **clé primaire**
- pour une **clé étrangère**
- pour un champ **"unique"**

2.5.2 Quels sont les avantages et les inconvénients des index ?

- Améliorations des performances de recherche

  Les index permettent de trouver plus rapidement les données en évitant de parcourir toute la table, surtout si elle contient beaucoup de lignes.

- Optimisation des requêtes

  Les index permettent au moteur de la base de données d'accéder directement aux données nécessaires, ce qui réduit le temps d'exécution, surtout pour les filtres (WHERE) ou les tris (ORDER BY).

- Ça prend de la place en mémoire

  Les index sont stockés à part dans la base de données, donc plus il y a d'index, plus ça consomme de l'espace disque et de mémoire.

- Ça ralentit les requêtes d'insertion, modification et suppression

  À chaque fois qu'on insère, modifie ou supprime une donnée, l'index doit être mis à jour, ce qui demande plus de temps et de ressources.

2.5.3 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.

2.6 Backup / Restore

2.6.1 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.

-u root : 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.

2.6.2 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.

-u root : 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.

3 CONCLUSION

Ayant déjà fait le module 106 et le projet lié, j'ai pu voir ma nette amélioration pour la compréhension et la mise en pratique. J'ai fait une énorme progression et même le professeur l'a remarqué. Il était plus simple pour moi de faire ce projet étant donné que je connaissais déjà le contenu, à quelques détails près. Hors ma progression, j'ai beaucoup apprécié réaliser ce projet, j'ai pu approfondir les connaissances que j'avais auparavant et apprendre de nouvelles choses.