# Langages d'exploitation des bases de données

Projet 1

# Énoncé

Vous devez concevoir et réaliser une base de données répondant à une problématique donnée.

Ce projet se veut un premier contact avec la conception et la réalisation d'une base de données. Pour y arriver, vous devrez réaliser 4 étapes distinctes, mais interdépendantes :

1. conception de la base de données;
2. réalisation des scripts permettant de créer les objets fondamentaux de la base de données :
    a. tables,
    b. contraintes;
3. réalisation des scripts permettant de populer la base de données;
4. réalisation des requêtes demandées.

# Problématique

## Mise en situation

Vous travaillez pour une entreprise en démarrage désirant s'établir dans le domaine du jeu vidéo. Vous avez rencontré des investisseurs, mais ces derniers restent frileux à cause de votre manque d'expérience. Pour les convaincre, vous décidez de mettre en place les l'essentiel d'une base de données permettant un fonctionnement rudimentaire de l'écosystème lié à votre projet : joueurs – avatars – jeux – activités.

Votre entreprise tente de percer le marché avec une idée originale consistant à pouvoir faire évoluer un avatar dans plusieurs mondes différents où chaque monde est un jeu différent. Ainsi, un avatar peut obtenir des habiletés exclusives à certains mondes, mais les réutiliser dans d'autres mondes. Cet aspect de l'écosystème garantit une expérience unique, car certaines habiletés disponibles dans certains mondes sont essentielles à la réussite d'objectifs présents dans d'autres mondes!

## Description sommaire du projet

L'approche consiste à développer plusieurs jeux de type MMORPG où vont évoluer les différents avatars. Chaque joueur (client) paye un tarif fixe mensuel et a accès à tous les mondes. Ainsi, un joueur inscrit qui a payé pour le mois courant peut jouer à n'importe quel jeu de l'écosystème avec chacun de ses avatars. Un joueur inscrit qui cesse de payer, peut à n'importe quel moment revenir et réactiver ses avatars dans l'état qu'ils avaient.

Chaque monde offre un nombre varié d'habilités exclusives où il est possible de les améliorer. De plus, lorsqu'une habileté est acquise, l'avatar peut l'utiliser dans tous les autres mondes. Évidemment, une habileté peut évoluer au fil du temps et devenir de plus en plus puissante, ce qui permet aux avatars d'obtenir un niveau de compétence lié indirectement à son temps de jeu.

Par exemple, les mondes suivants permettent les habilités suivantes :

- dans le jeu *SpaceX*™ :
    - ralentir le temps (amélioration possible : durée du ralentissement);
    - bouclier de proximité (amélioration possible: durée du bouclier);
- dans le jeu *DeepHorizonX*™ :
    - auto guérison (amélioration possible : capacité d'auto guérison plus fréquente);
    - implant d'un zoom oculaire (amélioration possible : niveau du zoom);

- dans le jeu *FreeZoneX*™ :
    - visée automatique (amélioration possible : précision);
    - ubiquité (amélioration possible: durée de la *multiprésence*).

Aussi, chaque monde présente des items rares et exclusifs. Il est possible de les obtenir à travers le jeu via diverses situations ou d'en acheter. Un avatar peut posséder plusieurs quantités du même item. Les items peuvent utilisés dans n'importe quel monde.

Dans le jeu, il existe une unité universelle d'échange utilisée pour différentes situations: le **moX**™. Les **moX** sont importants dans les jeux, car ils peuvent être utilisés pour améliorer une habileté, acheter des items ou tout simplement faire du troc entre ses avatars, mais aussi entre les avatars des autres joueurs.

## Données à traiter

Voici les données à manipuler :

- joueur : les joueurs sont les clients inscrits à notre service
    - alias — 32 caractères max, unique, obligatoire
    - courriel — 128 caractères max, unique, obligatoire
    - mot de passe — 32 caractères max, obligatoire
    - genre — une énumération entre f, h, x, NULL indique que le genre est inconnu, défaut NULL
    - date d'inscription — date, obligatoire, après le 1er janvier 2020
    - date de naissance — date, après le 1er janvier 1900 & le joueur doit avoir plus de 13 ans à l'inscription
- jeu : les jeux sont les différents logiciels supportant chacun des mondes
    - nom — 16 caractères max, unique, obligatoire
    - sigle — 6 caractères fixes, unique, obligatoire
    - description — 2048 caractères max
    - une liste de *n* habiletés que peut développer un avatar : — $n > 0$
        - nom — 32 caractères max, unique, obligatoire
        - sigle — 3 caractères fixe débutant toujours par 'S' majuscule, unique, obligatoire
        - énergie maximum — nombre variant de [10,.000 1000.000] (3 *digits*), obligatoire
        - $coef_1$, $coef_2$, $coef_3$ — nombre à double précision, défaut : 0, 0, 1, obligatoire
          ces trois paramètres déterminent la structure de coût selon le niveau :
          coût d'un niveau = $coef_1 \times niveau^2 + coef_2 \times niveau + coef_3$
        - description — 1024 caractères max
    - une liste de *n* items rares et exclusifs contenus dans le jeu : — $n \geq 0$
        - nom — 32 caractères max, unique, obligatoire
        - sigle — 4 caractères fixe débutant toujours par 'I' majuscule, unique, obligatoire
        - probabilité — nombre variant de ]0.000, 1.000[ (3 *digits*), défaut 0.025, obligatoire
        - description — 1024 caractères max
- avatar : les avatars sont les personnages virtuels créés par les joueurs
    - nom — 32 caractères max, obligatoire
    - *n* phrases personnelles (genre leitmotiv) — 64 caractères max chacune où *n* peut être 0
      par exemple : *I'm the boss – How do you feel now? – Don't cry!*
    - 3 couleurs préférées — encodé par un entier 32 bits, 1 obligatoire & 2 optionnelles
    - date de création — date, défaut date courante, obligatoire
    - quantité de **moX** — nombre entier variant de [-1e9, 1e9], défaut 0, obligatoire

    - pour chaque habileté développée :
        - date d'obtention de l'habileté — date et heure, défaut date et heure courante, obligatoire
        - niveau actuel — nombre variant de [1, 100], défaut 1, obligatoire
    - pour chaque item rare obtenu :
        - date d'obtention de l'item — date et heure, défaut date et heure courante, obligatoire
        - quantité — nombre variant de [1, 1e6], défaut 1, obligatoire
- activité : représente les informations générales de toutes les périodes d'activité des joueurs – par exemple, si le joueur se connecte et joue pendant 45 min le matin puis se reconnecte et joue pendant 3 heures le soir, il faudra enregistrer les informations liées à ces deux périodes d'activité
    - date et heure du début de l'activité — date, obligatoire
    - durée de l'activité — en seconde, doit être supérieure à 0, obligatoire
    - pour la durée de l'activité, on désire une liste de temps passé dans chaque monde pour chaque avatar joué – on nomme cette notion **capsule d'activité** — en seconde, > 0, obligatoire

Lorsque vous serez en industrie, ce genre de spécification correspondre à ce qu'un analyste logiciel peut produire.

Les données qui vous sont présentées sont considérées comme potentiellement incomplètes, fragmentées et non structurées pour une implémentation directe. Il est possible que l'ordre de présentation ne corresponde aucunement à la structure de votre base de données. C'est à vous d'interpréter et de transformer ces informations en modèle relationnel utilisable pour un vrai projet informatique de base de données.

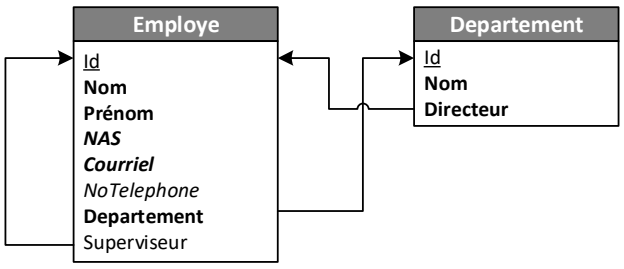
Finalement, votre enseignant a le rôle de client. Si vous avez des questions sur le projet et surtout pour clarifier des détails du mandat, c'est à lui qu'il faut s'adresser comme s'il était votre client. Cet énoncé présente plusieurs informations incomplètes et subtilités qu'il est possible d'interpréter différemment. Il est donc attendu que vous posiez des questions sur le projet.

# Les quatre tâches à réaliser

## 1. Conception

Vous devez prendre le temps de réfléchir à ce projet. Le défi est de le faire en équipe et de trouver une solution adéquate qui assure l'intégrité des données. Vous devez vous assurer de n'avoir aucune redondance d'information, c'est la partie la plus importante de tout le projet.

On vous demande de créer un graphique typique de base de données. Vous pouvez inventer votre propre notation pourvu que vous ajoutiez une légende explicative. Voici un exemple typique :




Exemple de schéma relationnel

Ce schéma est un bon exemple, car on vous demande d'ajouter clairement toutes les contraintes suivantes sur le graphique : clé primaire (soulignée), clé étrangère (illustré par une flèche directionnelle), unicité (en caractères italiques) et obligatoire (en caractères gras). Dans tous les cas, vous devez ajouter une légende explicative de votre notation.

Cette partie du travail, souvent sous-estimée et bâclée, est probablement la plus importante de tout le projet. Vous pouvez remettre un document réalisé avec un programme de dessin vectoriel comme Visio ou produit à la main (à condition qu'il soit propre, numérisé et intégré dans votre remise en version électronique – le document papier n'est pas à remettre).

## 2. Création des tables et contraintes

Vous devez créer un script (en format `pgsql`) permettant de créer automatiquement tous les objets fondamentaux de la base de données :

- tables
- contraintes (on vous demande de nommer les contraintes lorsque possible)

Votre script doit permettre de détruire les objets s'ils existent. N'oubliez pas qu'on vous demande d'ajouter toutes les contraintes de clé étrangères à la fin du script.

## 3. Insérer des enregistrements dans la base de données

On vous demande de rédiger un script permettant de populer les différentes tables avec des données pertinentes et cohérentes les unes avec les autres. Vous devez respecter ces nombreuses contraintes :

- 4 [3][1] joueurs différents (les joueurs doivent avoir vos noms) – un des joueurs sera désigné joueur principal (identifié par un ***** à la fin du nom) et contiendra plus de données que les autres pour les parties décrites plus bas;
- 1 avatar pour chaque joueur, le joueur principal doit en posséder 3 [2][1] dont un sera considéré principal (identifié par un ***** à la fin du nom de l'avatar);
- 3 [2][1] jeux ayant chacun 3 [2][1] habileté pour les avatars et un total de 6 [4][1] items rares;
- le joueur principal doit avoir 10 [8][1] capsule d'activités dont 8 [6][1] sont liés à l'avatar principal – les capsules d'activité doivent être distribuées en 4 [3] [1] activités différentes, tous les autres joueurs doivent avoir 3 [2][1] capsule d'activité distribuées sur un nombre d'activités laissé à votre discrétion;
- considérant toutes les autres tables (sauf pour les tables de liaison *n* vers *n*), vous devez insérer au moins 3 [2][1] entrées pertinentes et cohérentes avec le reste des données;
- pour les tables de liaisons *n* vers *n*, vous devez avoir les entrées nécessaires à la consistance de votre base de données considérant toutes les données présentes.

## 4. Rédaction de requêtes spécifiques

On vous demande de créer un script avec les requêtes répondant aux questions suivantes :

1. Donnez la liste de tous les joueurs en présentant : alias, courriel, date d'inscription.

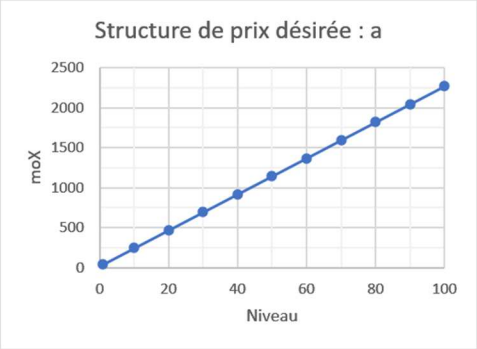
[1] Les indications entre crochets [*x*] représentent la quantité pour les groupes de taille réduite (3 personnes).

2. Donnez la liste des avatars d'un joueur quelconque (pour l'exemple, prendre le joueur principal), en donnant : nom, la couleur préférée transformée en trois composantes rouge-vert-bleu, date de création suivant le format 2000 | 12 | 25, le nombre de **moX**.
3. Pour l'avatar principal, donnez toutes les habiletés qu'il possède en présentant : le sigle et le nom entre crochets dans la même colonne, la date d'obtention, le niveau courant, la valeur en **moX** du niveau courant et le coût en **moX** pour acheter le prochain niveau.
4. Pour l'avatar principal, donnez la valeur totale des tous les items qu'il possède (les habilités considérant le niveau et les items considérant la quantité).
5. Pour le joueur principal, donnez le nombre total d'heures passées dans chaque jeu joué.
6. Donnez la liste de tous les avatars qui possèdent plus de 1 item : nom du joueur, nom de l'avatar et nombre d'items.
7. Chaque membre de l'équipe doit réaliser une requête **pertinente** de son cru. Considérant toutes les requêtes produites par l'équipe, vous devez respecter ces contraintes :
    a. il doit avoir au minimum 3, 4 et 5 tables dans l'une de vos requêtes,
    b. vous devez utiliser ces clauses: LIMIT, GROUP BY, ORDER BY, HAVING.

    À même le script et avant ces requêtes, vous devez ajouter un commentaire expliquant précisément ce que fait la requête. Le pointage associé à chacune de ces requêtes est autant basé sur la pertinence que la qualité de la réalisation technique.
8. [Boni] On désire modifier la structure de prix pour deux des items de façon à avoir les montants indiqués plus bas selon le niveau associé :

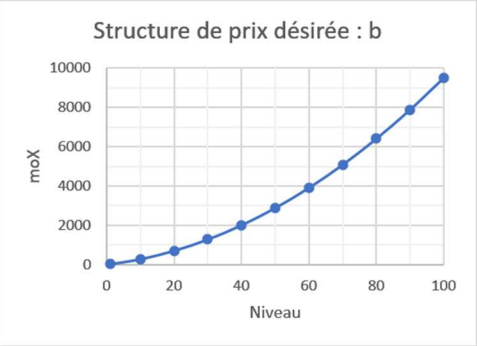
a.

| Niveau | moX |
|---|---|
| 1 | 42.5 |
| 10 | 245 |
| 20 | 470 |
| 30 | 695 |
| 40 | 920 |
| 50 | 1145 |
| 60 | 1370 |
| 70 | 1595 |
| 80 | 1820 |
| 90 | 2045 |
| 100 | 2270 |




b.

| Niveau | moX |
|---|---|
| 1 | 30.75 |
| 10 | 285 |
| 20 | 710 |
| 30 | 1285 |
| 40 | 2010 |
| 50 | 2885 |
| 60 | 3910 |
| 70 | 5085 |
| 80 | 6410 |
| 90 | 7885 |
| 100 | 9510 |




Pour chaque structure de prix, faites une requête permettant de modifier la structure de prix des items 1 et 2 que vous avez créée initialement.

Toutes vos requêtes doivent afficher un nom de colonne adéquat.

Finalement, toutes vos requêtes doivent avoir un commentaire d'en-tête présentant :

- No de la requête
- Énoncé de la requête
- Auteur et aide (entre parenthèse)
- Une ligne intitulée *Fonctionnelle* et indiquant :
    - oui (si la requête est complètement fonctionnelle)
    - aucunement (si la requête ne fonctionne pas du tout)
    - partiellement, *en ajoutant une description de ce qui ne fonctionne pas* (si la requête est partiellement fonctionnelle)

## Contraintes

Plusieurs contraintes sont à respecter pour ce travail :

1. Vous devez travailler en équipe de 4 ou de 3.
2. Il est très important que tous les membres de l'équipe travaillent équitablement, car l'évaluation finale tient compte de plusieurs critères, dont la répartition de la tâche de travail.
3. Vous devez travailler avec le SGBDR PostgreSQL utilisé tout au long de la session.
4. La date de remise est non négociable.

# Critères d'évaluation

1. Conception (30 points)
    - La conception répond à tous les critères demandés — 26 points
        - données bien structurée
        - contraintes
        - aucune redondance
        - répond à l'énoncé
    - Le schéma est propre et lisible — 2 points
    - Le schéma présente une légende — 2 points
2. Script de création (20 points)
    - La création des tables correspond au schéma — 7 points
    - Les types sont définis adéquatement — 4 points
    - Les contraintes sont pertinentes et répondent à l'énoncé — 5 points
    - Les contraintes de clé étrangère sont toutes ajoutées à la fin — 2 points
    - Les objets sont effacés au début du script — 2 points
3. Script d'insertion (20 points)
    - 4 [3] joueurs dont un joueur principal — 3 points
    - 4 [3] avatars dont un avatar principal — 3 points
    - 3 [2] jeux avec 3 [2] habiletés et 6 [4] items — 4 points
    - 19 [12] capsules d'activités — 4 points
    - autres tables (sauf *n* vers *n*) — 3 points
    - tables *n* vers *n* — 3 points
4. Requêtes (22 points)
    - Requête 1 — 1 point
    - Requête 2 — 2 points
    - Requête 3 — 2 points
    - Requête 4 — 2 points
    - Requête 5 — 2 points
    - Requête 6 — 3 points
    - Requête 7a (3 tables) — 2 points
    - Requête 7b (4 tables) — 2 points
    - Requête 7c (5 tables) — 3 points
    - Requête 7d (5 tables) – pour les équipes de3, le pointage est redistribué — 3 points
    - Requête 8 [boni] — *2 points
5. Qualité générale des scripts (8 points)
    - nom des tables, des colonnes et contraintes — 2 points
    - les en-têtes des scripts sont bien documentés — 2 points
    - les commentaires généraux sont adéquats — 2 points
    - respect général de la norme de codage — 2 points

# Stratégie d'évaluation

En cas de force majeur ou pour des raisons pédagogiques, la personne enseignante peut appliquer une note individuelle aux étudiantes et aux étudiants. La note peut, aussi, être basé sur les échanges et le travail en classe ou à l'extérieur (ex. Git ou message).

# Remise

Vous devez remettre les fichiers suivants en respectant les noms :

- votre schéma de conception `schema.pdf`
- le script de création de la base de données `creation.pgsql`
- le script permettant d'ajouter des données `insertion.pgsql`
- le script des requêtes demandées `requetes.pgsql`

Au final, vous devez remettre votre projet une seule fois sur le site Moodle du cours.