Groupe 6 :

Gestion de compte de Jeu de Cartes :

Dictionnaire de données

Description du monde réel :

Nous souhaitons modéliser une base de données capable de gérer les comptes des joueurs d'un jeu de cartes en ligne.

Chaque joueur possède un **Compte Joueur**, auquel est associé un pseudo, une adresse mail et un mot de passe. Il peut gagner de l'argent virtuel, en jouant, ce qui lui donnera une certaine quantité de monnaie appelée monnaieIG, soit en payant avec de l'argent réel, gagnant ainsi de la monnaieIRL. Au fil de ses combats, le joueur pourra gagner de l'expérience, puis, lorsqu'il en aura accumulé assez, gagner un niveau. De plus, chaque victoire lui permettra d'obtenir des points de division, lui permettant de se classer par rapport aux autres joueurs et d'atteindre des divisions de plus en plus prestigieuses. Enfin il pourra être **Ami** avec d'autres joueurs ou encore faire partie d'un rassemblement de joueurs, une **Guilde**, caractérisée par un nom unique, un niveau (ainsi que l'expérience qui va avec), et un nombre de places.

Le jeu étant un jeu de cartes, le joueur pourra (et devra) former des **Decks** (au nombre de deux) de **Cartes** à utiliser lors des combats. Une carte est associée à un nom, une rareté, un type, une description, une image, un coût en mana (pour l'utilisation lors des combats), une probabilité d'apparition après un match et un indicateur de possibilité d'amélioration.

Le joueur aura accès à une boutique permettant l'achat de divers produits présentés sous forme d'**Offres** :

- Les **Packs** : ce sont des paquets de cartes achetables en boutique. Ils permettent un certain nombre de pioches parmi des ensembles de certaines cartes choisis par les administrateurs. Les cartes piochées sont ainsi créditées au joueur.

- Les **Skin de Map** : ce sont des offres permettant de changer l'apparence de l'environnement lors des affrontements. Le joueur choisira et activera un SkinMap et un seul à la fois, parmi ceux qu'il possède.

- Les **Skin de Carton de Cartes** : ce sont des offres permettant de changer l'apparence des attributs visuels universels des cartes : contours, dos, etc. De même que pour les SkinMap, le joueur en activera un à la fois.

- Les **Icones de Joueurs** : ce sont des icônes que le joueur choisit pour personnaliser son compte. Le joueur en choisit une à la fois pour l'afficher.

- Les **Boosts** : ce sont des bonus permettant le gain de plus de monnaies, d'expériences, de cartes, etc, qu'un joueur active à l'achat, et qui dure pour une certaine durée, en nombre de matchs, ou en temps.

- L**'achat de monnaieIRL** : le joueur pourra utiliser sa carte bleue pour acheter une monnaie spéciale, la monnaieIRL, servant pour diverses transactions.

Enfin, le joueur pourra effectuer des Matchs, chacun dans un mode de jeu choisi par l'utilisateur, contre d'autres joueurs.

Réponses apportées au problème :

Structure de **CompteJoueur** :

Pour modéliser les comptes des joueurs, nous avons choisi d'utiliser une table **CompteJoueur** incluant les différents attributs ci-dessous :

Nom	Type	Description
Pseudo	varchar	Nom utilisateur associé au compte. **Clé primaire**.
mailCompte	varchar	Adresse email du compte. Unique (= un seul compte par adresse mail).
mdpCompte	varchar	Mot de passe du compte.
niveauJoueur	int	Niveau du compte.
expJoueur	int	Quantité de points d'expérience obtenus par le joueur.
ptsDivision	int	"Points de Division", représentative du classement du joueur et déterminant son appartenance à une certaine division.
monnaieIG	int	Quantité de monnaieIG que le joueur possède sur son compte.
monnaieIRL	int	Quantité de monnaieIRL que le joueur possède sur son compte

Reste alors l'appartenance à une division, une guilde et la liste d'amis. Cette dernière est une relation entre la table **CompteJoueur** et elle-même. Pour ce qui est de la division, on la modélisera par une table **Division**, possédant les attributs ci-dessous :

Nom	Type	Description
Id_Division	Int*	Identifiant unique de la division. **Clé primaire.**
nomDivision	varchar	Nom de la division.
iconeDivision	varchar	Fichier d'image de l'icône associé à la division.

Id_Division étant un clé étrangère placé dans **CompteJoueur** et référençant **Division**.

De même, la **Guilde** sera une table à part entière :

Nom	Type	Description
nomGuilde	varchar	Nom de la guilde. **Clé primaire** puisqu'unique.
niveauGuilde	int	Niveau de la guilde.
expGuilde	int	Quantité de points d'expérience obtenus par la guilde.
nbPlacesMax	int	Nombre de personnes maximal que la guilde peut accueillir à la fois.

NomGuilde étant une clé étrangère placé dans **CompteJoueur** et référençant **Guilde**.

Structure de **Cartes** :

Pour modéliser les cartes, on utilise une table **Cartes** possédant ces attributs :

Nom	Type	Description
Id_Carte**	Int*	Identifiant de la carte. **Clé primaire.**
NomCarte**	varchar	Nom de la carte.
rareteCarte	varchar	Rareté de la carte (commune, rare, épique …)
typeCarte	varchar	Type de la carte (sort, bombes, …)
descriptionCarte	varchar	Descriptif de la carte, affichable pour le client.
imageCarte	varchar	Fichier de l'image de la carte.
coutCarte	int	Cout en mana de la carte lors des combats.
DropRateCarte	int	Probabilité d'apparition d'une carte à la fin d'un combat.

estAmeliorable	boolean	Indicateur d'amélioration. Si X est améliorable, alors il existe une carte X' qui est une amélioration de X.

** : On n'utilise pas le nomCarte comme clé primaire puisqu'il peut exister plusieurs cartes ayant le même nom (améliorations, etc).

*** : stocké un int est un choix d'optimisation, s'il s'avère que le choix du traitement en objet n'est pas un bénéfice, il pourra etre possible de stocker des float plutôt que des int.

Constitution des **Decks** :

Un **Deck** est constitué de **Cartes** choisies par et pour un **CompteJoueur**.

Il existe donc une relation ternaire entre ces trois tables, que l'on nommera *JoueurCarteDeck*, ayant pour seul attribut qteCarte, déterminant le nombre d'exemplaires d'une même carte dans un deck d'un joueur.

Pour modéliser la construction des deux decks de chaque joueur, nous stockerons les cartes choisies dans deux lignes de **Decks**, deck1 et deck2, qui seront nommés [Pseudo][1-2].

Pour modéliser la possession d'une carte X par un joueur, nous stockons la carte X dans le deck0, nommé [Pseudo][0].

Ainsi, le principe est que pour chaque joueur, il existe 3 decks de cartes :

- Le deck 0, qui est l'ensemble des cartes que possède le joueur;
- Et les decks 1 et 2, qui sont les deux ensembles de cartes que le joueur a choisi pour combattre.

Tout sera donc stocké dans la relation JoueurCarteDeck. La table **Deck** est donc simplement de la forme :

Nom	Type	Description
Id_Deck	varchar	Identifiant du Deck. Toujours de la forme [PseudoDuJoueur]{0,1,2}. **Clé primaire**.

Principe de la Boutique et des différents produits :

Les différents produits sont stockés selon ces différents schémas :

- **IconeJoueur** :

Nom	Type	Description
Id_IconeJoueur	Int*	Identifiant de l'icône. **Clé primaire**.
nomIcone	varchar	Nom de l'icône.
imageIcone	varchar	Fichier d'image de l'icône.
imageMiniatureIcone	varchar	Image de la miniature présentant l'icône en boutique.
descriptionIcone	varchar	Descriptif de l'icône auprès du client.

- **SkinMap** : Cette table est à-même de changer selon l'avancement de l'interface utilisateur ***lors des combats*** et donc de la portée des changements apportés par une apparence personnalisée.

Nom	Type	Description
Id_SkinMap	Int*	Identifiant du Skin. **Clé primaire**.
nomMap	varchar	Nom du SkinMap.
imageFondMap	Varchar	Fond d'écran lors des combats.
imageTableMap	varchar	Apparence de la table lors des combats.
imageMiniatureMap	varchar	Image de la miniature présentant le SkinMap en boutique.
descriptionMap	varchar	Descriptif de l'apparance de la map auprès du client.

- **SkinCartonCarte** : Cette table est à même de changer selon l'avancement de l'interface utilisateur ***lors des combats*** et donc de la portée des changements apportés par une apparence personnalisée.

Nom	Type	Description
Id_SkinCartonCarte	Int*	Identifiant du Skin. **Clé primaire**.
nomCarton	varchar	Nom du carton.
imageVersoCarte	varchar	Image du verso des cartes lors des combats.
imageContourCarte	varchar	Image du contour des cartes lors des combats.
imageMiniatureCarton	varchar	Image de la miniature présentant l'apparence des Cartes en boutique.
descriptionCarton	varchar	Descriptif de l'apparence des cartes auprès du client.

- **Boost** :

Nom	Type	Description
Id_Boost	int	Identifiant du boost. **Clé primaire**.
nomBoost	varchar	Nom du boost.
typeBoost	varchar	Type du boost. Permet de connaître la durée et le type de bonus.
imageMiniatureBoost	varchar	Image de la miniature présentant le boost en boutique.
descriptionBoost	varchar	Descriptif du boost auprès du client.

- **Pack** :

Le principe voulu du pack est de permettre un tirage aléatoire dans un paquet de cartes. Cependant, on aimerait pouvoir limiter le nombre de tirages (et donc de cartes obtenus), mais aussi de forcer l'aléatoire pour ne pas entrainer de frustration auprès de l'utilisateur (acheter un pack "Pack avec une grande chance d'obtenir des rares !" n'impose pas forcément d'obtenir des cartes rares, ce qui peut être une mauvaise chose). On décomposera ainsi les Packs en 3 couches :

- Les Packs eux-mêmes : ce sont les boites noires qu'achèteront les clients.

Nom	Type	Description
Id_Pack	Int Auto-increment	Identifiant du Pack. **Clé primaire**.
nomPack	varchar	Nom du Pack.
imageMiniaturePack	varchar	Image de la miniature présentant le Pack en boutique.
descriptionPack	varchar	Descriptif du Pack auprès du client.

- Les LootPacks : ce sont les objets qui permettent de forcer l'aléatoire. Chaque Pack est composé d'un ou plusieurs LootPacks, et va tirer dans chacun d'eux un certain nombre de fois, déterminé par un attribut qteCarte, représentant le nombre de cartes que l'on va tirer depuis le LootPack. Ainsi, la somme des qteCarte est égale au nombre de cartes dans le Pack.

Nom	Type	Description
id_LootPack	Int Auto-increment	Identifiant du LootPack. **Clé primaire**.

qteCarte	int	Nombre de cartes que l'on tire dans le LootPack [id_LootPack].

En toute logique, il existe une relation entre LootPack et Pack. On estime qu'un LootPack n'appartient qu'à un et un seul Pack, ce qui traduit la présence d'une clé étrangère dans LootPack référançant Pack.

- Les Ensembles de cartes : ce sont les objets qui permettent le tirage aléatoire. Ces Ensembles sont constitués de cartes. Chaque LootPack contient un ou plusieurs Ensembles, ayant chacun une probabilité de tirage, qui déterminera la probabilité que l'on a, depuis un LootPack, de piocher dans un certain Ensemble.

Nom	Type	Description
Id_Ensemble	Int Auto-increment	Identifiant de l'ensemble. **Clé primaire**.
dropRatePack	int	Probabilité d'aller piocher dans l'ensemble [id_Ensemble].

Ainsi, il existe une relation LootPack et Ensemble, ainsi qu'une autre entre Ensemble et Carte. On estime qu'un Ensemble peut appartenir à plusieurs LootPacks.

Finalement, avec une telle structure, il est possible de créer n'importe quel pack, puisqu'on choisit le nombre de cartes et dans quelle mesure leurs tirages seront aléatoires.

L'achat d'un de ces produits a un effet parmi trois possibles :

- Si l'on achète un **Pack**, les cartes obtenus sont ajoutés au Deck 0 du joueur.
- Si l'on achète une **Icone** (resp. un **SkinMap**, un **SkinCartonCarte**), alors celui-ci est ajouté à une relation entre **IconeJoueur** (resp. **SkinMap**, **SkinCartonCarte**) et **CompteJoueur**, nommé *posséderIconeJoueur* (resp. *posséderSkinMap*, *posséderSkinCartonCarte*). Cela implique que l'achat d'un de ces produits soit unique.
- Si l'on achète un **Boost**, cela l'active pour une certaine durée, en nombre de matchs (nbMatchsFin) ou en heures (HeuresFin) pour le joueur. On l'ajoute donc à une relation *activer* entre **CompteJoueur** et **Boost**.

Le dernier point à éclaircir est le choix de l'Icone, du SkinMap et du SkinCartonCarte utilisés. Ce choix est modélisé par une relation entre CompteJoueur et l'Icone (resp. SkinMap, resp. SkinCartonCarte). Un CompteJoueur ne pouvant utiliser qu'une Icone (resp. SkinMap, resp. SkinCartonCarte) à la fois, ces relations impliquent la présence de clés étrangères dans CompteJoueur référençant chacun l'Icone (resp. SkinMap, resp. SkinCartonCarte) choisi(e).

Tout ces différents produits sont proposés dans des Offres. Une Offre se compose ainsi :

Nom	Type	Description
Id_Offre	Int Auto-increment	Identifiant de l'Offre. **Clé primaire**.
nomOffre	varchar	Nom de l'offre.
prixMonnaieIG	int	Prix de l'offre en monnaie IG.
prixMonnaieIRL	int	Prix de l'offre en monnaie IRL.
typeOffre	varchar	Définit ce que l'on vend dans l'offre.
ImageOffre**	varchar	Image de l'offre.

** : L'image est utilisée lorsque l'offre est un ensemble de plusieurs offres, sinon, on utilise simplement la miniature de l'objet vendu.

Cette table est en relation avec tout les produits et selon certaines propriétés dépendant du produit :

- On estime qu'il n'y a qu'un pack par Offre, puisqu'on peut "facilement" créer un Pack équivalent à l'union de plusieurs autres Packs.
- On estime qu'il peut y avoir plusieurs icones/SkinMap/SkinCarton/Boost par Offre.
- On estime que l'on peut combiner différents produits dans une même offre, ce qui revient à vendre une Offre de type "Mix" (nom susceptible de changer).

Tout cela implique donc la présence d'une relation par produit, et d'une clé étrangère d'Offre vers Pack.

Enfin, pour la dernière fonctionnalité, l'**achat de monnaieIRL**, on permettra un suivi de ces achats plus spéciaux que les autres, via une relation *HistoriqueAchatIRL*, entre **CompteJoueur** et **AchatMonnaieIRL**, ayant pour attribut dateAchatIRL, indiquant la date de la transaction.

Nom	Type	Description
Id_AchatIRL	int Auto-increment	Identifiant de la transaction. **Clé primaire**.
prixEuros	int	Prix en euros de la transaction. Une simple conversion permettra de l'afficher dans une autre monnaie, on choisit arbitrairement l'euros.
gainMonnaie	int	La quantité de monnaieIRL obtenu suite à la transaction.

Gestion des Matchs :

Enfin, pour gérer les **Matchs** des joueurs, nous utiliserons une table **Matchs** :

Nom	Type	Description
Id_Match	Int Auto-increment	Identifiant du match. **Clé primaire**.
debMatch	TIMESTAMP	Date de début du match.
finMatch	TIMESTAMP	Date de fin du match. Null si le match est en cours.

Matchs possède trois relations : deux vers **CompteJoueur**, *JoueurDansMatch* et *Historique*, et une vers **ModeDeJeu**.

JoueurDansMatch définit qui sont les deux joueurs du match, créant deux clés étrangères dans **Matchs** référençant **CompteJoueur**.

Historique liste les **Matchs** effectués par un **joueur**. Elle possède un attribut *vainqueur* qui est le pseudo du joueur victorieux.

La relation vers **ModeDeJeu** ne sert qu'à référencer le Mode de Jeu de la partie, créant une clé étrangère vers **ModeDeJeu**.

Le **mode de jeu** est simplement modélisé par une table de la forme :

Nom	Type	Description
Id_ModeDeJeu	Int	Identifiant du Mode de jeu. **Clé primaire**.
nomModeDeJeu	Varchar	Nom du Mode de jeu.
descriptionModeDeJeu	Varchar	Descriptif du Mode de jeu auprès du joueur.

---

* : Certaines de nos clés primaires de type Entier ne sont pas en auto-increment. Ce choix s'explique principalement par sécurité vis-à-vis de notre traitement **en objet** des informations mis en base de données : en effet, nous estimons qu'un traitement en objet par le serveur sera plus rapide qu'une requête SQL à la base de données, et afin d'être le plus optimisé possible et d'éviter au maximum les temps de latence pendant les affrontements, nous estimons que ce choix est un bon début.

Ce choix nous force à effectuer un traitement spécifique en fonction de l'identifiant. En toute logique, lors d'un combat, le traitement d'une carte X ne sera pas le même que pour une carte Y. Ainsi, si par malheur nous devions réinsérer des objets dans un autre ordre que celui de base, on fausserait totalement certains de nos traitements par le serveur.

Ainsi, pour garder un contrôle sur nos objets, nous contrôlons nous-même nos identifiants, ce qui nous permettra aussi de pouvoir insérer les objets peu importe l'ordre.

On notera aussi que certains de nos attributs seront peut être amenés à devenir Auto-increment, tout comme d'autres pourront ne plus l'être.