[1]*Travail final d'intégration*

Projet, StarLab !

- Ce travail sera réalisé **individuellement**
- À terminer pour le vendredi 10 décembre 2021, 23h59
- Le travail compte pour 25 % de la note finale;

Objectifs

Ce travail vise principalement à vous faire expérimenter les aspects suivants:

- Écrire des procédures stockées;
- Écrire des déclencheurs;
- Expérimenter l'aspect sécurité des données;
- Intégration avec un langage de programmation;
- Veiller à ce que les données soient cohérentes en tout temps (transactions)
- Explorer un SGBD NoSQL, MongoDB

Mise en situation

Nous sommes sur le e vaisseau spatial « *Patrouilleur* » qui est un vaisseau de découvertes d'autres planètes et d'autre espèces. L'équipage de ce vaisseau est assez important et spécialisé. Chaque groupe de spécialiste a des privilèges sur les bases ou les bases de données du vaisseau.

Sur ce vaisseau, nous avons deux départements distincts : Le département « Exploration et développement » (**EDEV**) dont le responsable est **James Kirk** et le département «éQuipage et Gestion» (**QGES**) dont la responsable est **Rachel Tilly**

Dans ce vaisseau, nous avons les classes :

- La classe des ingénieurs, il y a 3 ingénieurs : James Kirk, Thomas Trip et Melinda Tores.
- La classe des officiers tactiques : Spoky Spoke, Lanna Tipole
- La classe des enseignes, nous retiendrons : Rachel Tilly, Samuel Wiky, Saturne Lune et Legrand Sarru
- La classe des médecins : Lapointe Hugh,
- La classe cuisiniers : Lewis Nellis

[1] Source de l'image : https://pixabay.com/fr/illustrations/vaisseau-spatial-star-trek-5181695/

Authentification au serveur MS SQL

Les tâches spécifiques à la gestion de « MS SQL Server » telles que créer et supprimer une base de données, gérer les rôles et autorisations, créer les connexions et les usagers des bases de données ainsi que de créer les tables dans la base de données **dbges** seront réservées uniquement à l'administrateur du serveur qui est **Alain Patoche, qui est le commandant du vaisseau.**

Pour vaquer à ses activités, l'**administrateur** s'authentifiera au serveur avec la connexion **vos_initiales_dba** (par exemple : **SY_dba** – vous pouvez utiliser une authentification SQL Server pour **vos_initiales_dba).**

Chaque membre de l'équipage aura sa propre connexion pour s'authentifier au SGBDR. Ces connexions n'auront aucun autre rôle **serveur** que **public**.

Finalement, le jeu **Trivial Pursuit** aura aussi sa propre connexion, **trivial,** qui sera utilisée par l'interface graphique pour se connecter au SGBDR. Cette connexion n'aura aucun autre rôle serveur que **public**.

Les bases de données du vaisseau

Chacun des départements possède sa propre base de données. Les données propres au département des «éQuipage et Gestion» (**QGES**) sont conservées dans la base de données **dbges** et celles propres au département « Exploration et développement » (**EDEV**) dans la base de données **dbdev**.

À leur création, le nom de ces bases de données doit-être précédé de vos initiales (exemple : **SYdbges**)

Base de données dbges

La base de données **dbges** contiendra les 4 tables décrites ci-dessous.

Classes	Équipages	Explorations	Missions
classeno(pk)	equipano (pk)	equipano (fk)	idMission (pk)
nomClasse	nom	idMission(fk)	nomMission
salaireMin	prenom	nbPoints	dateMission
salaireMax	adresse		niveauRisque
	salaire		
	classeno (fk)	**(**equipano **i**dMission**)** **forment une PK**	

Toutes les primary key ont un type **int**. equipano et idMission sont IDENTITY dans leurs tables d'origine. (ne peuvent pas être IDENTITY dans Explorations)

Les salaires ont un type **money; nbPoints** int

La dateMission a un type **date.**

Les autres colonnes sont des **varchar (30).**

niveauRisque prend ses valeurs dans : élevé, moyen, faible

nbPoints est 1000 pour une mission à risque faible, 1500 pour une mission à risque moyen et 5000 pour une mission à haut risque. (fait par la procédure stockée ***insererExploration***)

Le salaire de l'équipage est contrôlé par un trigger ***CTRLSalaire*** qui garantit que le salaire d'un membre de l'équipage est à l'intérieur de sa classe. Si le salaire est à l'extérieure des bornes (salaireMin, salaireMax) l'insertion et les mises à jour sont annulées

Initialement, la table Equipages contient tous les membres de l'équipage du vaisseau

Initialement, la table Classes contient

idClasse	nomClasse	salaireMin	salaireMax
1	Commandant	200 000	500 000
2	Ingénieurs	150 000	300 000
3	Officiers tactiques	180 000	350 000
4	Enseigne	50 000	120 000
5	Médecin	200 000	450 000
6	Cuisinier	45 000	90 000

Le Holo Deck 1 et la Base de données dbdev

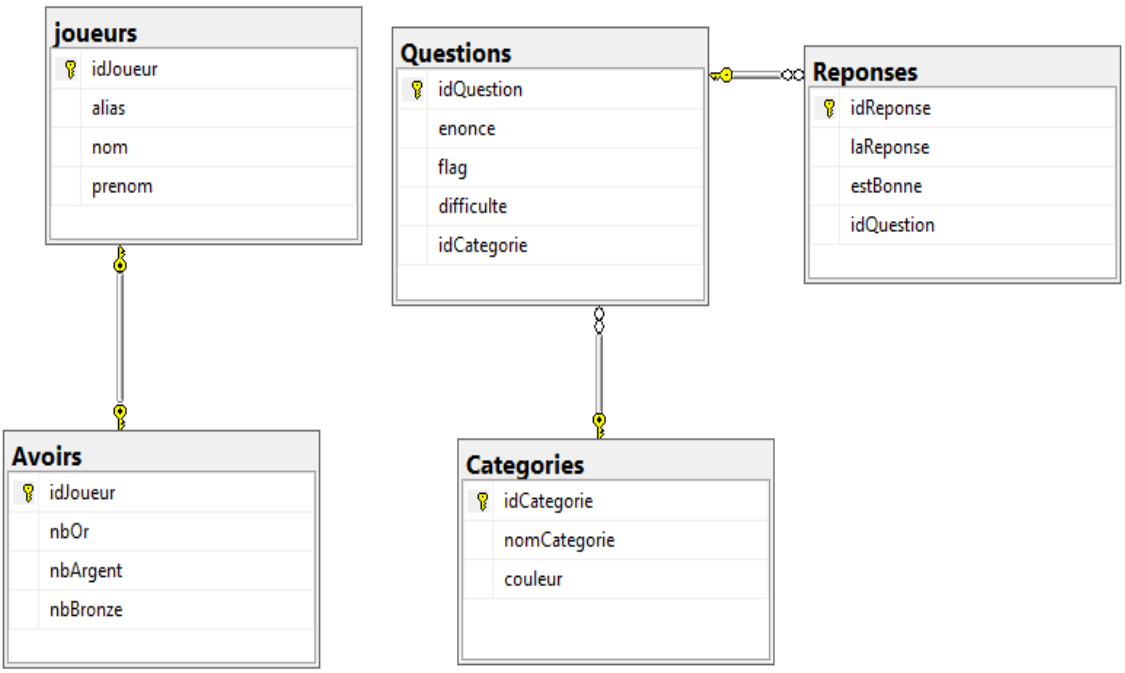
Les ingénieurs et les officiers tactiques du vaisseau voulant se détendre durant leurs heures libres jouent au jeu **Trivial Pursuit**, jeu de questions et réponses que nous avons utilisé pour quelques démonstrations du cours.

La base de données **dbdev** contiendra les tables du jeu Trival Pursuit (**voir le script)** et ainsi que la table **Avoirs** décrite ci-dessous.

Les joueurs dans la table joueurs sont les ingénieurs et les officier tactiques du vaisseau

Colonnes	Description
idJoueur (pk) et fk	Référence à un joueur de la table joueur
nbOr	Nombre de pièces d'or
nbArgent	Nombre de pièces d'Argent
nbBronze	Nombre de pièces de bronze

Voici la partie du modèle de du jeu Trivial Pursuit qui nous intéresse (la table joueurs est liée à d'autres tables qui ne sont pas sur le modèle)



Résumé : Voici comment l'équipage du vaiseau « *Patrouilleur* » est organisé

Sécurité et intégrité des données au QGES

Pour respecter les règles de sécurité, l'administrateur (DBA) Alain Patoche décide de ne pas donner tous les droits à tous les membre de l'équipage. Il décide plutôt d'y aller par attribution de rôles. Pour chaque classe d'équipage, un rôle est défini.

Voici un tableau qui résume un peu la structure du « *Patrouilleur* » concernant les droits et les privilèges.

Roles	Privilèges	Membres
RoleTrivial	SELECT, UPDATE, INSERT sur toutes les tables de la base de données **dbdev**	trivial
RespQges	SELECT, UPDATE, INSERT sur toutes les tables du département QGES. Également le privilège EXECUTE.	Responsable QGES
RoleEnseigne	SELECT sur toutes les tables du département QGES, UPDATE de la table Equipages, UPDATE de la colonne niveau de risque dans la table Missions INSERT et UPDATE de la table Explorations	Enseignes
RoleCuisinier	SELECT sur toutes les tables du département QGES	Le Cuisinier
db_owner	Propriétaire de la base de données du département EDEV	Responsable du département EDEV
db_datawriter	Peuvent écrire sur la BD du département EDEV	Les ingénieurs et les officiers tactiques
db_datareader	Peuvent lire sur la BD du département EDEV	Les ingénieurs et les officiers tactiques
db_ddladmin	Peuvent exécuter les commandes DDL su la bd du département EDEV	Les ingénieurs et les officiers tactiques
En plus, Les ingénieurs et les officiers tactiques **doivent pouvoir exécuter les procédures stockées de la base de données de leur département**		

Tableau 1 :Tableau 1 : Rôles et autorisations

Ce qu'on vous demande :

1. Créer les bases de données des deux départements, QGES et EDEV
2. Créer les tables pour le département QGES, puis insérer les données.
3. Écrire la procédure stockée : ***insererExploration*** qui permet de faire une insertion dans la table Explorations
4. Écrire la fonction table ***totalPoints*** qui retourne pour chaque membre de l'équipage la somme des points acquis pour les explorations.
5. Exécuter le script TrivialPursuit.sql pour créer les tables du département EDEV et y insérer les données
6. Créer les logins avec les utilisateurs mappés sur les logins dans la base de données correspondante. (Les ingénieurs , les officiers tactiques et le responsable de EDEV sont

rattachés à la bd **dbdev**. Les enseignes, le cuisinier et le responsable QGES sont rattachés à la base de données **dbges**)

7. Attribuer les rôles serveur et bases de données s'il y a lieu
8. Créer les RÔLES non prédéfinis du tableau 1
9. Ajouter les membres aux rôles en tenant compte des autorisations.
10. Faire en sorte que les ingénieurs et les officiers tactiques puissent exécuter les procédures stockées.
11. Programmer le trigger qui contrôle le salaire de l'équipage.

Développement et sécurité des données au département EDEV

Les ingénieurs et les officiers tactiques du vaisseau voulant se détendre durant leurs heures libres jouent au jeu **Trivial Pursuit**, jeu de questions et réponses que nous avons utilisé pour quelques démonstrations du cours.

A- Système de récompense, et questions réponses

Dans le jeu Trivial Pursuit, chaque question a un niveau de difficulté de 1 à 3 : 1 pour facile, 2 pour moyen et 3 pour difficile. La quantité de pièces obtenues en fonction du niveau de difficulté de la question est affichée dans le tableau ci-dessous.

Niveau de difficulté	Nombre de pièces obtenues
1	5 pièces de bronze
2	10 pièces d'argent
3	15 pièces d'or

Les récompenses ainsi obtenues seront comptabilisées dans la table **Avoirs** (ci-dessus) par la procédure **validerRéponse**.

Ce qu'on vous demande

1. Écrire la procédure stockée **PigerQuestion** qui permet de piger une question de manière aléatoire à partir de la table questions et de mettre le flag de la question à « o » (comme quoi la question a été pigée).

2. Écrire une procédure **ValiderRéponse** qui comptabilise les pièces d'or, d'argent ou de bronze associées à la question. Les récompenses sont comptabilisées dans la table **Avoirs**. En d'autres mots, lorsqu'un joueur (alias) répond correctement à une question (a la bonne réponse) alors la table Avoirs est mise à jour selon le degré de difficulté de la question.

3. Programmer la procédure **AjouterQuestionRéponses** qui permet d'ajouter une question avec ses réponses associées;

4. Pour garantir l'intégrité des données de la table **réponses** et s'assurer qu'il n'existe en tout temps qu'une seule bonne réponse par question, vous devez programmer le trigger **CTRLRéponse.**

5. L'information de la table **Avoirs** est confidentielle. Il est donc d'une importance capitale qu'un usager de la base de données du département EDEV ait accès uniquement aux données qui le concernent. Ici on suppose qu'un joueur correspond à un usager. Vous devez mettre en place un mécanisme qui garantira :

    a. Qu'un joueur puisse consulter ses avoirs et uniquement les siens
    b. Qu'un joueur puisse mettre à jour (INSERT et UPDATE) ses avoirs et uniquement les siens.

B- Authentification des joueurs et information de crédit

Des projets sont en discussion pour implémenter un système d'achats en ligne pour les joueurs inscrits au jeu. L'aspect sécurité des données devient donc un enjeu important.

Dans un premier temps, vous devez implémenter un mécanisme d'authentification des joueurs à l'aide d'un mot de passe. Le mot de passe des joueurs sera conservé dans la table joueurs et sera chiffré à l'aide d'une fonction de hachage. Vous aurez donc à ajouter deux nouvelles colonnes à la table **joueurs** pour conserver ce mot de passe et les informations de crédit.

De plus, les joueurs auront la possibilité d'associer un numéro de carte de crédit à leur profil dans la table **joueurs**. Votre travail est d'ajouter la fonctionnalité pour pouvoir conserver en toute sécurité le numéro de carte de crédit des joueurs.

Le numéro de carte de crédit sera crypté à l'aide d'un algorithme de chiffrement symétrique. La clé utilisée pour crypter le numéro sera le mot de passe du joueur. Cette façon de procéder garantit qu'uniquement le joueur en possession du mot de passe aura accès à l'information de crédit.

Ce qu'on vous demande

Pour mettre en place cette nouvelle fonctionnalité, vous devez programmer les procédures et fonctions suivantes.

1. Programmez la procédure **AjouterJoueur** qui ajoute un joueur à la table joueurs avec toutes ses informations, incluant son mot de passe.
2. Programmez la procédure **ModifierMotPasse** qui permet à un joueur de modifier son mot de passe. L'ancien mot de passe doit être fourni pour valider son identité.
3. Programmez la fonction scalaire **ValiderIdentité** pour valider l'identité d'un joueur basé sur son mot de passe. La fonction retourne 0 (succès) ou 1 (échec).
4. Programmez la procédure **AjouterInfoCrédit** qui permet d'ajouter un numéro de carte de crédit à un joueur donné. Le mot de passe du joueur doit être fourni.
5. Programmez la fonction table **ObtenirInfoCrédit** qui retourne toute l'information d'un joueur, incluant son numéro de carte de crédit, mais pas son mot de passe.
6. Pour chaque procédure, donnez au moins une exécution qui marche et une exécution qui ne marche pas.

C- ADO.Net

Vous devez concevoir une petite application (C#/ADO.NET/Form) pour permettre l'ajout de questions et réponses dans la base de données du jeu Trivial Pursuit.

L'interface utilisateur doit permettre de saisir un **alias** et un mot de passe. Cette information servira à authentifier le joueur. Pour authentifier le joueur, vous utiliserez les services que vous avez implémentés dans la section précédente

Seulement après avoir été authentifié avec succès, la fonctionnalité d'ajout de questions/réponses sera activée.

Le logiciel devra utiliser la connexion **triviale** pour s'authentifier au SGBDR.

Le Holo Deck, Humanoïde

Pour se détendre, les enseignes du vaisseau jouent au jeu « les monstres de l'espace », ***mongoStat*** permet de garder les statistiques des joueurs, Pour chaque joueur, nous souhaitons stocker, puis afficher :

- Les créatures méchantes qu'il a combattues (orc, orgre, sangliers méchant, renard à 4 têtes, ogopogo, etc.), la force de la créature (fort, moyen et faible).
- Les créatures amies (Phénix, Harpie etc…) qu'il a apprivoisées.
- Les points cumulés. Pour les points, lorsque le joueur tue une créature méchante :
  - Forte, il gagne 1 5 points;
  - De force moyenne, il gagne 10 points;
  - De force faible il gagne 5 points.
- Lorsque le joueur apprivoise une créature amie, il gagne 20 points.
- Les joueurs ont un id, et un alias et pourraient avoir d'autres informations

À titre d'indication, voici un exemple de document que vous pouvez utiliser.

- Les joueurs sont : Rachel Tilly, Samuel Wiky, Saturne Lune et Legrand Sarru
- Les créatures peuvent être de votre choix

```
{
    "_id": 201,
    "alias":"Tilly",
    "creatures" :
            {"nom":"Elfe",
             "type":"gentil",
             "nbPoints": 20
            }
}
```

Ce qu'on vous demande

1. Créer la base de données MongoDB ***mongoStat***
2. Créer la collections Joueurs
3. Dans la collection joueurs, y insérer les joueurs qui sont des enseignes.
4. Écrire l'application C# pour permet :
   a. D'insérer un document
   b. De rechercher les joueurs qui ont confronté une créature donnée. L'affichage se fera dans une liste

Ce que vous devez remettre :

1. Un fichier SQL portant votre nom, identifié à l'intérieur par votre nom et contenant :
    - La création des BD
    - La création des tables
    - Les insertions initiales
    - Les requêtes SQL bien identifiées par leurs numéros et la section.
    - Les triggers identifiés incluant les tests d'exécution
    - Les procédures (fonctions) stockées identifiées par **leurs noms et la section. Incluant les tests d'exécution**

2. Un projet Visual Studio (ADO.NET/Form)
3. Le fichier txt contenant le code ***mongoStat***
4. L'application C# ***mongoStat***

Modalité de remise : Boite de remise

Vous allez remettre 4 fichiers clairement identifiés par VOS NOMS

- Le projet ADO.NET, zippé
- Un fichier SQL.
- Fichier mongoStat, au format txt
- Le projet C# mongoStat

Barème :

Éléments évalués	Sur
Sécurité et intégrité des données au QGES	30
Développement et sécurité des données au département EDEV	52 (total = 20+20+12)
A- Système de récompenses et questions réponses	20
B- Authentification des joueurs et information de crédit	20
C- ADO.NET	12
Le Holo Deck , Humanoïde (mongoStat)	8
Respect des consignes de remise : ici vous avez 10 ou zéro, Si une consigne n'est pas respectée alors vous avez 0 . il est important que votre code soit clair. Que vos requêtes SQL ne fassent pas 1 mètre, que chaque procédure soit identifiée par sa section et son numéro.	10
Total	100