# [1]Production finale d'intégration : Sur le vaisseau « patrouilleur »

- Ce travail sera réalisé **individuellement**
- À terminer pour jeudi 14 décembre minuit.
- Le travail compte pour 20 % de la note finale;
- Le plagiat sous, toute ses formes, entraine les sanctions prévues par la PIEA

## 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;
- Veiller à ce que les données soient cohérentes en tout temps (transactions)
- Introduction à MongoDB;

## Mise en situation

Nous sommes sur le 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 la base ou les bases de données du vaisseau.

Sur ce vaisseau, nous avons deux départements distincts : Le département « Exploration et développement » (**Exp**) dont le responsable est Kathryn Janeway et le département «Quartier Général» (**Qgs**) dont le responsable est Chakotay Pierre

Dans ce vaisseau, nous avons les classes :

- La classe des commandants : Kathryn Janeway, Chakotay Pierre
- La classe des ingénieurs , il y a 3 ingénieurs : Tom Paris, Thomas Trip et Melinda Tores.
- La classe des officiers tactiques : Touky Tuvok, Seven ofNine
- La classe des enseignes, nous retiendrons : Rachel Tilly , Samuel Wiky, 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 .**On considère :

- Que votre compte admin joue le rôle du compte Alain Patoche
- La connexion au serveur est une connexion avec authentification SQL Server.

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 **TrouverRelique** aura aussi sa propre connexion, **Relic,** qui sera utilisée par l'interface graphique(qui ne sera pas développée ici) pour se connecter au SGBDR. Cette connexion n'aura d'autres rôles que **public**.

## Les bases de données du vaisseau

Chacun des départements possède sa propre base de données. Les données propres au département du «Quartier Général » (**Qgs**) sont conservées dans la base de données **dbges** et celles propres au département « Exploration et développement » (**Exp**) dans la base de données **dbexp**.

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 | Equipages | Explorations | Missions |
|---|---|---|---|
| classeno(pk) | equipano (pk) | equipano (fk) | idMission (pk) |
| nomClasse | nom | idMission(fk) | nomMission |
| | prenom | nbPoints | niveauRisque |
| | race | dateExploration | |
| | classeno (fk) | **(**equipano **i**dMission**) forment une PK** | |

1. Toutes les primary key ont un type INT. equipano et idMission sont IDENTITY dans les tables d'origine.
2. La dateExploration a un type date. Les autres colonnes sont des varchar .
3. niveauRisque prend ses valeurs dans : Élevé, moyen, faible
4. race :peut être humain, vulcain, Borg etc…
5. nbPoints est 1000 pour une mission à risque faible, 1500 pour une mission à risque moyen et 5000 pour une mission à haut risque. nbPoints de type INT
6. Les cuisiniers ne peuvent pas faire de missions
7. Les médecins ne peuvent faire de missions dont le niveau de risque est élevé.

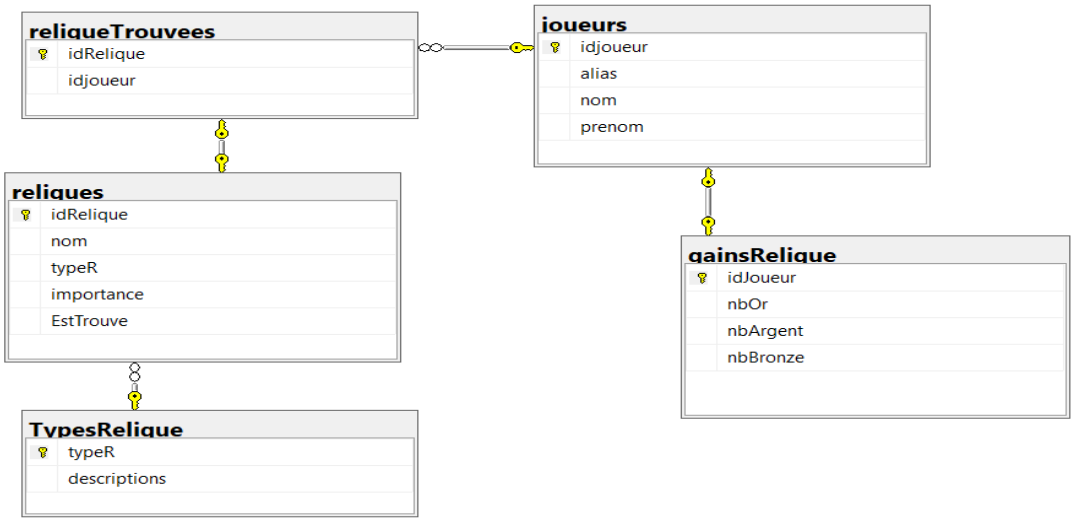
- Initialement, la table Equipages contient tous les membres de l'équipage du vaisseau
- Initialement, la table Classes contient :

| idClasse | nomClasse |
|---|---|
| 1 | Commandant |
| 2 | Ingénieurs |
| 3 | Officiers tactiques |
| 4 | Enseigne |
| 5 | Médecin |
| 6 | Cuisinier |

- Pour la table missions, vous pouvez saisir les données de votre choix.

## Le Holo Deck 1 et la nostalgie du temps passé : La Base de données dbexp

Les ingénieurs et les officiers tactiques du vaisseau voulant se détendre durant leurs heures libres ont développé le jeu **TrouverRelique**. Un jeu qui envoi l'équipage dans un temps très loin dans le passé. Ce jeu est une chasse aux reliques décrit par le modèle de données suivants :




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

## Sécurité et intégrité des données au QGS :

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.

| Rôles | Privilèges | Membres |
|---|---|---|
| **RoleRelic** | SELECT, UPDATE, INSERT sur toutes les table de la base de données **dbExp**. | relic |
| **RespQgs** | SELECT, UPDATE, INSERT sur toutes les tables du département QGS. Également le privilège EXECUTE. | Responsable Qgs |
| **RoleEnseigne** | SELECT sur toutes les tables du département Qgs,<br>UPDATE de la table Equipages,<br>UPDATE de la colonne niveau de risque dans la table Missions<br>INSERT et UPDATE de la table Explorations | Enseignes, le médecin |
| **RoleCuisinier** | SELECT sur toutes les tables du département Qgs | Le Cuisinier |
| **db_owner** | Propriétaire de la base de données du département Exp | Responsable du département Exp |
| **db_datawriter** | Peuvent écrire sur la BD du département Exp | Les ingénieurs et les officiers tactiques |
| **db_datareader** | Peuvent lire sur la BD du département Exp | Les ingénieurs et les officiers tactiques |
| **db_ddladmin** | Peuvent exécuter les commandes DDL sur la bd du département Exp | 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 pour cette partie :

1. Créer les bases de données des deux départements, Qgs et Exp
2. Créer les tables pour le département QGS, puis insérer les données.
3. Écrire la procédure stockée : ***insererExploration*** qui permet de faire une insertion dans la table Explorations;
4. Écrire le trigger qui garantit : lors des insertions dans la table Explorations si le nombre de points ne correspond pas à la difficulté de la mission alors l'opération est annulée.
5. Écrire le trigger qui garantit les points 6 et 7 du carré vert plus haut. (classe VS la mission)
6. Écrire la fonction table ***totalPoints*** qui retourne pour chaque membre de l'équipage la somme des points acquis pour les explorations.
7. Exécuter le script ***TrouverRelique.sql*** pour créer les tables du département Exp (dbExp) et y insérer les données
8. Créer les logins avec les utilisateurs mappés sur les logins dans la base de données correspondante. Les logins et les user seront sous ce format : Première lettre du prénom suivit du nom (sans espaces, sans accent. Exemple : syacoub). Vous devez respecter la stratégie des mots de passe.
9. Attribuer les rôles serveur et bases de données s'il y a lieu
10. Créer les RÔLES non prédéfinis du tableau 1
11. Ajouter les membres aux rôles en tenant compte des autorisations.
12. Faire en sorte que les ingénieurs et les officiers tactiques puissent exécuter les procédures stockées.

Le tout doit être remis dans un fichier sql : *pfi_dbqges.sql*

# Développement et sécurité des données au département EXP

## A- Développement au département EXP : Le Holo Deck 1

Les ingénieurs et les officiers tactiques du vaisseau ayant développé le jeu **TrouverRelique** veulent effectuer les tests nécessaires en jouant au jeu**.** Ce qui leur permet également de se détendre durant leurs heures libres.

### Description du jeu TrouverRelique :

Les joueurs ont pour but de trouver des reliques cachées sur le site de la ville de Kent. Les reliques sont uniques. (deux joueurs ne peuvent pas trouver la même reliques).

Les reliques ont une propriété : **estTrouve** qui indique si la relique est trouvée ou non (1→ la relique est trouvées, 0-→ la relique n'est pas trouvées). Initialement toutes les reliques ne sont pas trouvées.

Les reliques ont une importance qui est 1, 2 ou 3.

- Lorsqu'un joueur trouve une relique d'importance 1, on lui donne 5 pièces de Bronze
- Lorsqu'un joueur trouve une relique d'importance 2, on lui donne 10 pièces d'Argent
- Lorsqu'un joueur trouve une relique d'importance 3, on lui donne 15 pièces de d'Or.

Ces données sont consignées dans la table **`gainsRelique`**

**Lorsqu'un joueur trouve une relique, les opérations suivantes doivent être effectuées et la BD doit rester dans un état cohérent:**

- On doit mettre à jour que la relique est trouvée
- On attribue la relique au joueur en question. Ces informations sont dans la table : **`reliqueTrouvees`**
- On donne le nombre de pièces correspondant (Or, Argent, Bronze) au joueur correspondant. La table affectée est **`gainsRelique.`**
- Si le joueur existe dans la table **`gainsRelique,`** on met à jour ses gains sinon on fait une insertion.

### Ce qu'on vous demande

1. Écrire la procédure stockée ***ajouterRelique*** qui permet d'ajouter une relique dans la base de données

2. Écrire une procédure ***objetTrouve(@alias varchar(10), @idRelique int)*** qui permet de trouver une relique. Cette procédure permet de faire les opérations décrites dans l'encadré vert de la page 5. De plus cette procédure doit vérifier que l'alias existe, et que la relique n'est pas encore trouvée.

3. Programmer une procédure stockée ***classementJoueurs*** affiche le classement des joueurs. On affiche l'alias, et le nombre de pièces associées à leurs trouvailles. Les joueurs ayant le nombre total de pièces d'Or en premiers, puis, Argent, enfin Bronze

4. Écrire une procédure ***updateRelique*** qui met le flag estTrouve à 0 lorsque toutes les reliques sont trouvées.

5. L'information de la table **`gainsRelique`** est confidentielle Il est donc important qu'un usager de la base de données du département Exp 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 gains et uniquement les siens
    b. Qu'un joueur puisse mettre à jour (INSERT et UPDATE) ses gains et uniquement les siens.

    **Pour la question précédente, on vous demande de faire les tests pour deux joueurs uniquement.**

## 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 (**ENCRYPTBYPASSPHRASE**). 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.

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.

## C- Le Holo Deck 2 , Humanoïde

Au niveau de ce Holo Deck, les membres de l'équipage 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 joueurs ont un id, et un alias et pourraient avoir d'autres informations
- Chaque créature apprivoisée ou combattue donne un certain nombre de points

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

```
{
    "_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 :
    a. Un joueur à la fois;
    b. Deux ou trois joueurs à la fois
4. Écrire la requête qui cherche :
    a. Les joueurs selon leur alias .
    b. Les joueurs qui ont apprivoisé des Elfes
5. Écrire l'application qui utilise un langage de haut niveau (**C# console** )qui permet :
    a. D'insérer un document
    b. De rechercher les joueurs qui ont confronté une créature donnée.

## Ce que vous devez remettre :

- Un script (pfi_dbqges.sql) contenant les commandes sql de la partie « Sécurité et intégrité des données au QGS »
- Un script (pfi_dbExp.sql), spécifique au département EXP, les réponses aux questions de la partie Développement au département Exp et au jeu TrouverRelique
- Un script (pfi_dbExpsecure.sql), spécifique au département Exp, contenant la partie « authentification des joueurs » au département Exp.
- Un script (pfi_mongostat.js ou txt) contenant les requêtes du jeu du Holo Deck 2.
- L'application C# mongostat au format zippé

## Modalité de remise : Boite de remise

## Barème :

| Éléments évalués | Sur |
|---|---|
| Sécurité et intégrité des données au QGS | 36 |
| Développement et sécurité des données au département EXP | 54 =(20+24 +10) |
| A- Développement au département EXP : (Holo Deck 1 et TrouverRelique) Q1/3 , Q2/6, Q3 /3, Q4/3, Q5/4 | 24 |
| B- Authentification des joueurs et information de crédit (Holo Deck 1) | 20= 4*5 |
| C- Le Holo Deck 2 , Humanoïde (mongoStat) | 10 |
| Respect des consignes de remise | 10 |
| Total | 100 |