**Module : 62-31 – Gestion avancée des données**
**Sujet : Examen du mercredi 18 janvier 2023**

h e g | Haute école de gestion Genève, Informatique de gestion

# Modalités

a) Durée : 2h (14h30 – 16h30)
b) Documentation personnelle autorisée.
c) Interdiction de poser des questions sur des forums ou semblables.
*(ChatGPT étant un outil fournissant des réponses, il ne reflète pas votre compétence de résolution de problème. Son utilisation n'est pas souhaitée car elle peut être assimilée à une demande adressée à une tierce personne).*
d) Toute communication *(ou tentative de communication)* en ligne ou en direct sera sanctionnée par la note de 1.
e) Téléchargez le fichier **TransportsEntreAgences.zip** depuis Cyberlearn : en haut du **Cours 62-31**.
f) Décompressez le zip contenant le script CreerEnv-Exa.sql et le projet IntelliJ à compléter.
g) Exécutez le script CreerEnv-Exa.sql qui crée les tables et les données initiales.
h) **Reddition** : à la fin de votre travail, zippez le tout *(scripts pl/sql + projet)* puis déposez-le sur Cyberlearn :

**Examen 62-31 TransportsEntreAgences – Rendu**

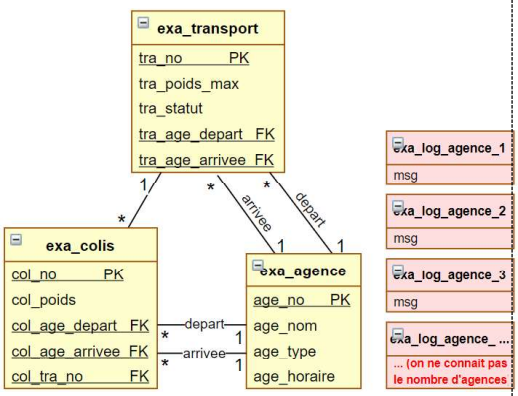
# Énoncé :

Une société possédant plusieurs agences à Genève *(Carouge, Meyrin, Lancy, …)* effectue fréquemment des trajets entre ses agences pour transférer des colis.

Elle possède une base de données Oracle contenant la liste des colis à transporter d'une agence à l'autre, une base de données MongoDB où elle conserve les transports effectués, et une base de données Neo4j contenant les trajets existants entre agences *(navettes)* permettant d'optimiser ses transports.

Elle vous demande donc de gérer correctement les nouveaux transports, et rajouter des validations afin d'éviter de modifier ou supprimer les transports déjà effectués.

X? tables de log *(une par agence)* permettent d'indiquer les colis qui n'ont pas été transportés *(dans le cas où le poids dépasserait le maximum)*.




# Vue à disposition : vw_exa_transports contenant les données suivantes :

| | NUMERO | STATUT | POIDS_MAX | DEPART | AGENCE_DEPART | ARRIVEE | AGENCE_ARRIVEE | NO_COLIS | POIDS_COLIS |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | T | 50 | 1 | Point de vente de Carouge | 3 | Point de retrait de Corsier | 104 | 25 |
| 2 | 1 | T | 50 | 1 | Point de vente de Carouge | 3 | Point de retrait de Corsier | 101 | 10 |
| 3 | 2 | T | 10 | 3 | Point de retrait de Corsier | 5 | Point conseil de Lancy | | |
| 4 | 3 | P | 25 | 2 | Point de retrait de Bernex | 1 | Point de vente de Carouge | | |
| 5 | 4 | P | 25 | 1 | Point de vente de Carouge | 3 | Point de retrait de Corsier | 110 | 10 |
| 6 | 5 | P | 50 | 2 | Point de retrait de Bernex | 4 | Point de vente de Meyrin | 113 | 20 |
| 7 | 5 | P | 50 | 2 | Point de retrait de Bernex | 4 | Point de vente de Meyrin | 111 | 10 |

L'application Java n'a pas accès directement aux tables, elle passera toujours par la vue ci-dessus pour accéder aux données *(lecture, modification, suppression)* stockées dans Oracle.

# Fonctionnalités souhaitées, travail demandé :

## a) Insertion d'un nouveau transport *(à faire sur le serveur, donc en PL/SQL !)*

- Lorsqu'une application souhaite insérer un nouveau transport, elle est obligée de passer par la vue et exécute donc l'instruction suivante :

```
INSERT INTO vw_exa_transports (poids_max, depart, arrivee) VALUES (20, 1, 3)
```

- Les 3 seules valeurs à fournir sont le poids maximum total pour ce transport *(somme des poids des colis maximum, sinon le colis n'est pas transporté et inscrit dans le log)*, le n° d'agence de départ et d'arrivée.
- Un trigger dans Oracle s'occupera donc d'ajouter correctement ce nouveau transport dans la base de données, avec la liste des colis qui seront transportés *(modification de la fk exa_colis.col_tra_no)*.

- Les règles à respecter sont les suivantes :
  - inutile de s'occuper du numéro du transport, une séquence `sq_exa_transport_no` s'occupe déjà de les numéroter automatiquement *(c'est déjà fait, vous n'avez pas de nextval à faire !)*.
  - le `statut` du nouveau transport sera mis obligatoirement à `'P'` *(pour Prêt au transport)*.
  - les colis à prendre en compte dans ce transport sont ceux dont le n° d'agence de départ et d'arrivée correspondent aux numéros fournis, et qui n'ont pas encore de n° de transport défini.
  - si aucun colis ne satisfait les conditions, aucun transport ne sera créé !
  - s'il y a moins de 10 colis à transporter *(de l'agence de départ à l'agence d'arrivée, et qui n'ont pas de n° de transport défini)*, alors on prend les premiers par ordre de poids *(du plus lourd au plus léger)*, sinon on les prend dans l'ordre de leur n° de colis *(du plus petit au plus grand n° de colis)*.
  - on prend les colis jusqu'à concurrence du poids maximum indiqué : une variable poids_total sera incrémentée pour chaque colis ajouté dans le transport. Il ne faut pas dépasser le poids maximum.
  - les colis qui seront donc transportés seront modifiés : on indique leur n° de transport dans la table.
  - les colis qui ne sont pas transportés *(sinon on dépasserait le poids maximum défini)* seront répertoriés dans le log de l'agence *(table Oracle **`exa_log_agence_X`**, **`X`** étant le n° d'agence de départ)* avec le message d'information suivant : « `Le colis n°114 n'a pas été pris !` »
- Tous ces traitements seront évidemment faits sur le serveur, c'est-à-dire en PL/SQL !

## b) Modification et suppression d'un transport *(instructions lancées en Java, gestion en pl/sql)*

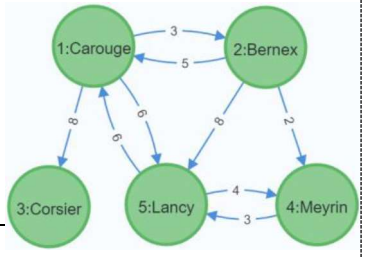
- Les applications *(par exemple en Java)* doivent avoir la possibilité de modifier ou supprimer un transport existant, avec les restrictions suivantes :
  - Les mises-à-jour s'effectueront généralement via la vue `vw_exa_transports`.
  - Si le transport est terminé (`statut = 'T'`), seul le champ `statut` pourra être modifié.
    Les modifications des autres champs seront simplement ignorées, sans indication supplémentaire, même si on essaie de modifier un champ de la table sans passer par la vue !
  - Il doit être possible de supprimer un transport *(toujours via la vue, à l'aide de l'instruction `DELETE vw_exa_transports WHERE …`)*, mais seuls les transports qui ne sont pas terminés (`statut` différent de `'T'`) pourront être supprimés, sinon le message d'erreur suivant sera retourné à l'application : « `Impossible de supprimer le transport n°2 (déjà effectué !)` ».

## c) Conserver les transports dans MongoDB *(à faire dans l'application Java)*

- La méthode `copierLeTransportDansMongoDb` doit récupérer les données du transport stockées dans Oracle via la vue `vw_exa_transports`, puis créer un nouveau document dans MongoDB dans une collection nommée « `Transport` » avec les informations suivantes :
  - Le n° du transport, et le poids maximum autorisé.
  - Les données de l'agence de départ et d'arrivée *(une collection MongoDB « `Agence` » existe !)*.
  - La liste de tous les colis de ce transport *(numéro et poids)*.
- Vous devez bien évidemment structurer correctement ce nouveau document dans MongoDB !

## d) Afficher le chemin *(la distance)* pour aller de l'agence de départ à l'agence d'arrivée

- La base de données **Neo4j** contient la liste des agences, avec les distances des trajets entres elles.
- Si une navette directe existe, on affiche la distance (1 --> 2 ==> distance = 3).
- S'il y a un chemin qui existe, mais qui passe par d'autres agences, on le spécifie (1 --> 4).
- S'il n'y a aucune possibilité d'aller de l'agence de départ à l'agence d'arrivée, on le mentionne (3 --> 5).
- Et on s'assure que les numéros d'agence fournis sont corrects (8 --> 9) !




- Le jeu de test fournit doit afficher les résultats suivants :

```
1 --> 2  ==> Le trajet de "Carouge" vers "Bernex" = 3
1 --> 4  ==> Le trajet de "Carouge" vers "Meyrin" passe par d'autres agences !
3 --> 5  ==> Le trajet de "Corsier" vers "Lancy" n'existe pas !
8 --> 9  ==> Agence inconnue !
```