TP Noté - Langage SQL et SGDB SQL Server

Gaël Roustan (Argonaultes)

2026

Abstract

Ce document contient les instructions du devoir noté.

Human Bot

Contexte

Votre société de conseil idBOT est appelée pour moderniser l'approche métier d'une entreprise familiale 'HUMAN BOT' spécialisée dans la création de robots humanoïdes.

Toute la société était gérée sur papier. Votre objectif est de remplacer cette gestion papier par une gestion entièrement numérique avec notamment une base de données SQL Server accompagnée des procédures et fonctions adéquates pour la communication avec les futures applications nécessitant un échange d'information avec cette nouvelle base de données.

Consignes et évaluations

Il faudra donc respecter à la lettre les noms des fonctions, procédures et triggers que l'on vous demande de créer.

Ces procédures, fonctions et triggers sont nécessaires dans la mesure où les applications n'auront jamais connaissance de la structure des tables, qui pourra évoluer au fil du temps sans que cela ait un impact sur les applications tierces.

Livrable

A l'issue de votre session de travail, vous devez fournir plusieurs fichiers SQL à travers un repo GIT dont vous fournirez l'adresse via le formulaire Google. Ce repo GIT doit contenir les fichiers suivants avec les noms exacts :

- `schema.sql` : fichier SQL contenant la définition du schéma avec les contraintes
- `data.sql` : fichier SQL contenant les requêtes d'insertion de données
- `views.sql` : fichier SQL contenant les vues
- `functions.sql` : fichier SQL contenant les fonctions
- `procedures.sql` : fichier SQL contenant les procédures
- `triggers.sql` : fichier SQL contenant les triggers

Le non respect de cette consigne entrainera une perte automatique de 4 points.

Argonaultes, RCS PARIS 978 665 909, NAF : 85.59A
SARL au capital de 3 000 euros,
enregistré sous le numéro 11756794275 auprès du préfet de région d'Ile-de-France

Détails du fonctionnement (pour la création des tables)

Créer votre propre structure et ajouter toutes les contraintes métier que vous jugerez pertinentes sur la base de la conversion entre un consultant de votre société de conseil idBot et un gestionnaire de l'entreprise 'HUMAN BOT'.

Voici une restranscription de l'interview des gestionnaires principaux de l'entreprise.

Consultant : Combien d'usines devez-vous gérer ?

Gestionnaire : Nous avons plusieurs usines qui fonctionnent presque sur le même modèle. C'est à dire que nous avons presque les mêmes informations sur le personnel intervenant sur les chaines de production. Pour l'usine à Paris, nous avons les nom, prénom et age. Pour l'usine à Caracas, nous n'avons que le nom et prénom des intervenants. En revanche, nous avons à chaque fois la date de début et la date de fin de contrat.

Consultant : Est-ce qu'il est possible qu'une personne intervienne sur plusieurs usines et/ou pour la même usine dans le cadre de plusieurs contrats ?

Gestionnaire : Oui

Consultant : Combien de temps avez-vous besoin de garder les informations ?

Gestionnaire : Si au bout de 5 ans l'intervenant n'a pas fait l'objet d'un nouveau contrat, nous supprimons sa fiche.

Consultant : Souhaitez-vous gérer d'autres éléments ?

Gestionnaire : Le suivi des pièces détachées et la quantité de productions de chaque usine.

Consultant : Vous ne crééz pas toutes vos pièces ?

Gestionnaire : Non, nous devons acheter toutes nos pièces détachés à différents fournisseurs. Nous pourrions être perçus d'ailleurs plutôt comme des usines d'assemblage plutôt que des usines de fabrication.

Consultant : Le même fournisseur vous fournit toutes vos pièces ?

Gestionnaire : Non pas du tout, nous avons plusieurs fournisseurs qui nous fournissent différentes pièces de chaque robot. Chaque robot peut être vu comme un ensemble de pièces à assembler et nous devons nous assurer qu'une usine a bien reçu l'ensemble des pièces pour pouvoir assembler le robot demandé.

Argonaultes, RCS PARIS 978 665 909, NAF : 85.59A
SARL au capital de 3 000 euros,
enregistré sous le numéro 11756794275 auprès du préfet de région d'Ile-de-France


Consultant : Et vous souhaiteriez relancer un fournisseur possédant la pièce manquante par exemple ?

Gestionnaire : Oui exactement, en fonction des prévisions de production, nous aimerions ajuster notre carnet de commandes. J'allais oublier, nous avons préparé quelques données avec des exemples de réponse attendues pour vous donner un exemple de traitement de données.

Vues

1. Vue `ALL_WORKERS` qui affiche en lecture seule pour l'ensemble des employés les propriétés suivantes :

- `lastname`
- `firstname`
- `age`
- `start_date`

Trier le résultat afin que le salarié arrivé le plus récemment soit affiché en premier. Si des valeurs sont manquantes, convservez les tout de même dans le résultat. N'affichez que les travailleurs toujours présents dans l'usine.

2. Vue `ALL_WORKERS_ELAPSED` qui donne le nombre de jours écoulés depuis l'arrivée de chaque employé au moment de la requête en vous basant sur la vue `ALL_WORKERS`.

- `lastname`
- `firstname`
- `nb_days_elapsed`

3. Vue `BEST_SUPPLIERS` qui affiche les fournisseurs ayant livré plus de 1000 pièces. Trier le résultat afin que le fournisseur ayant livré le plus grand nombre de pièces apparaisse en premier.

- `supplier`
- `nb_parts`

4. Vue `ROBOTS_FACTORIES` qui permet de connaître l'usine ayant assemblé chaque robot enregistré.

- `factory`
- `nb_robots`


Fonctions

La définition des fonctions demandées doit être enregistrée dans le fichier `functions.sql` à la racine de votre repo GIT.

1. Créer la fonction `GET_NB_WORKERS(FACTORY NUMBER) RETURN NUMBER` qui renvoie le nombre de travailleurs dans une usine fournie en paramètre en utilisant l'une des vues fournies.
2. Créer la fonction `GET_NB_BIG_ROBOTS RETURN NUMBER` qui compte le nombre de robots assemblés avec plus de 3 pièces en utilisant l'une des vues fournies.
3. Créer la fonction `GET_BEST_SUPPLIER RETURN VARCHAR2(100)` qui renvoie le nom du meilleur fournisseur basée sur la vue `BEST_SUPPLIERS` en utilisant l'une des vues fournies.
4. Créer la fonction `GET_OLDEST_WORKER RETURN NUMBER` qui renvoie l'identifiant du travailleur le plus ancien parmi tous les travailleurs de toutes les usines en utilisant l'une des vues fournies.

Procédures

La définition des procédures demandées doit être enregistrée dans le fichier `procedures.sql` à la racine de votre repo GIT.

1. Créer la procédure `SEED_DATA_WORKERS(NB_WORKERS NUMBER, FACTORY_ID NUMBER)` qui crée autant de workers que demandé. Les valeurs pour le nom seront `'worker_f_' || id` et `'worker_l_' || id` et la date de démarrage sera prise au hasard entre le 01/01/2065 et 01/01/2070.
2. Créer la procédure `ADD_NEW_ROBOT(MODEL_NAME VARCHAR2(50))` qui crée un nouveau modèle depuis la vue `ROBOTS_FACTORIES`.
3. Créer la procédure `SEED_DATA_SPARE_PARTS(NB_SPARE_PARTS NUMBER)` qui créé autant de spare parts qu'indiqué en paramètre.

Triggers

La définition des triggers demandés doit être enregistrée dans le fichier `triggers.sql` à la racine de votre repo GIT.

1. En cas de tentative d'insertion d'un worker dans la vue `ALL_WORKERS_ELAPSED`, intercepter la demande d'insertion pour effectuer l'insertion dans la bonne table. Dans les autres cas, `UPDATING` et `DELETING`, lever une erreur.
2. A chaque nouveau robot créé, enregistrer la date d'ajout du robot dans la table `AUDIT_ROBOT`.


3. Si le nombre d'usines dans la table `FACTORIES` n'est pas égal au nombre de table respectant le format `WORKERS_FACTORY_<N>` alors empêcher toute modification de données via la vue `ROBOTS_FACTORIES`.
4. En cas d'ajout de date de départ pour un worker, calculer la durée du temps passé dans l'usine donnée et stocker la valeur dans une nouvelle colonne à ajouter au schéma si elle n'existe pas encore dans la table.