Gestion d'un service de réparation d'objets

Requêtes, vues, triggers et données

Vitória Cosmo, Aubry Mangold, Eva Ray

14 décembre 2023

Requêtes

Les requêtes suivantes ont été créées afin de manipuler la base de données :
— Consulter les réparations pour un technicien donné.
— Modifier l'état d'une réparation.
— Modifier la description d'une réparation.
— Ajouter un temps de travail à une réparation.
— Consulter les données d'un client.
— Modifier les données d'un client.
— Consulter les données d'une réparation.
— Créer une réparation.
— Modifier les données d'une réparation.
— Modifier l'état du devis d'une réparation.
— Consulter les données d'une vente.
— Créer une vente.
— Modifier les données d'une vente.
— Envoyer un SMS.
— Modifier l'état d'un SMS.
— Consulter un SMS.
— Consulter les données d'un échange SMS lié à une réparation donnée dans l'ordre antéchronologique.
— Modifier les données d'un collaborateur.
— Effacer une personne.
— Effacer une vente.
— Dans l'API, des requêtes basiques pour mettre en place les opérations CRUD (SELECT, INSERT, UPDATE, DELETE) sur les différentes tables ont été créées.

Les requêtes suivantes ont été réalisées à de fins statistiques :
— Obtenir la quantité d'employés par rôle.
— Obtenir la quantité de réparations en cours.
— Obtenir la quantité de réparations par catégorie.
— Obtenir la quantité de réparations terminées.
— Obtenir la quantité de réparations terminées par catégorie.
— Obtenir la quantité d'objets en vente.
— Obtenir la quantité d'objets vendus.
— Obtenir la quantité d'objets traités par catégorie.
— Obtenir la quantité d'objets traités par marque.
— Obtenir la quantité d'heures travaillées par spécialisation.
— Obtenir la quantité de réparations traitées par mois.
— Obtenir la quantité de réparations créées par réceptionniste.
— Obtenir la quantité de réceptionnistes par langue.
— Obtenir la quantité de SMS reçus par jour.
— Obtenir la quantité de SMS envoyés par jour.

Vues

Les vues suivantes ont été créées :
- `collab_role_id_view` : affiche les id des collaborateurs et leur rôle.
- `works_on` : liste les réparations en cours et le temps travaillé pour un technicien donné.
- `technician_view` : affiches les informations des réparations auquel un technicien est lié.
- `collab_info` : affiche les infos d'un collaborateur.
- `receptionist_view` : affiche les infos d'un réceptionniste.
- `customer_info_view` : affiche les infos d'un client.

Triggers

Les triggers suivants ont été créés :
- `on_reparation_update_update_date` : met à jour le champ `date_modified` à la date actuelle lors d'une mise à jour d'une réparation.
- `verify_quote_state_is_declined_for_reparation` : vérifie que le champ `quote_state` est bien à `declined` lors d'une insertion dans la table `sale`.
- `verify_quote_state_is_declined_for_object` : vérifie que le champ `quote_state` est bien à `declined` lorsque le champ `location` est mis à jour en `for_sale` ou `sold`.
- `verify_quote_state_is_accepted` : vérifie que le champ `quote_state` est bien à `accepted` lorsque le champ `reparation_state` de réparation est mis à jour en `ongoing` ou `done`.
- `verify_tos_accepted` : vérifie que les `tos` sont bien acceptés lors de la création d'une réparation.
- `set_reservation_state_ongoing` : met à jour le champ `reparation_state` en `ongoing` lorsque le champ `quote_state` de la réparation est mis à jour en `accepted`.
- `before_insert_receptionist_language` : si un langage n'est pas présent dans la table `language`, lors d'une insertion dans la table de jointure `receptionist_language`, le langage est inséré dans la table `language`.
- `update_collaborator_person_trigger` : update les tables `collaborator` et `person` lors d'une mise à jour de la view `collab_info_view`.
- `update_collaborator_person_trigger_on_insert` : update les tables `collaborator` et `person` lors d'une insertion dans la view `collab_info_view`.
- `delete_collaborator_person_trigger_on_delete` : supprime les lignes des tables `collaborator` et `person` lors d'une suppression dans la view `collab_info_view`.«<
- `update_customer_person_trigger` : update les tables `customer` et `person` lors d'une mise à jour de la view `customer_info_view`.
- `update_customer_person_trigger_on_insert` : update les tables `customer` et `person` lors d'une insertion dans la view `customer_info_view`.
- `delete_customer_person_trigger_on_delete` : supprime les lignes des tables `customer` et `person` lors d'une suppression dans la view `customer_info_view`.
- `update_collaborator_person_receptionist_trigger` : update les tables `collaborator`, `person` et `receptionist` lors d'une mise à jour de la view `receptionist_view`.
- `update_collaborator_person_receptionist_trigger_on_insert` : update les tables `collaborator`, `person` et `receptionist` lors d'une insertion dans la view `receptionist_view`.
- `delete_collaborator_person_receptionist_trigger_on_delete` : supprime les lignes des tables `collaborator`, `person` et `receptionist` lors d'une suppression dans la view `receptionist_view`.
- `update_collaborator_person_technician_trigger` : update les tables `collaborator`, `person` et `technician` lors d'une mise à jour de la view `technician_view`.
- `update_collaborator_person_technician_trigger_on_insert` : update les tables `collaborator`, `person` et `technician` lors d'une insertion dans la view `technician_view`.
- `delete_collaborator_person_technician_trigger_on_delete` : supprime les lignes des tables `collaborator`, `person` et `technician` lors d'une suppression dans la view `technician_view`.
- `update_collaborator_person_manager_trigger` : update les tables `collaborator`, `person` et `manager` lors d'une mise à jour de la view `manager_view`.
- `update_collaborator_person_manager_trigger_on_insert` : update les tables `collaborator`, `person` et `manager` lors d'une insertion dans la view `manager_view`.
- `delete_collaborator_person_manager_trigger_on_delete` : supprime les lignes des tables `collaborator`, `person` et `manager` lors d'une suppression dans la view `manager_view`.

Procédures

Les procédures suivantes ont été créées :
— `InsertCustomer` : insère un client dans la base de données.
— `InsertCollaborator` : insère un collaborateur dans la base de données.
— `InsertManager` : insère un manager dans la base de données.
— `InsertReceptionist` : insère un réceptionniste dans la base de données.
— `InsertTechnician` : insère un technicien dans la base de données.
— `createReceptionist` : crée un nouveau réceptionniste (utilisée par l'api)
— `updateReceptionist` : met à jour un réceptionniste (utilisée par l'api)
— `createReparation` : crée une nouvelle réparation (utilisée par l'api)

Données

Des données ont été insérées dans toutes les tables de la base de données. Les points suivants sont relevants :
— 50 clients, 3 managers,15 techniciens et 3 réceptionnistes sont insérés.
— 50 réparations et objets ont été insérées. Chaque réparation est liée à un client différent. Les différents états des énumérations `quote_state`, `reparation_state` et `location` sont utilisés.
— 100 SMS ont été insérés. Chaque réparation est liée à un couple de SMS. Les SMS sont tous dans l'état `processed`.
— 6 ventes ont été insérées. Certaines des ventes sont terminées.
— Les spécialités, catégories, marques, languages et autres tables sont remplis.