**Laboratoire 05**

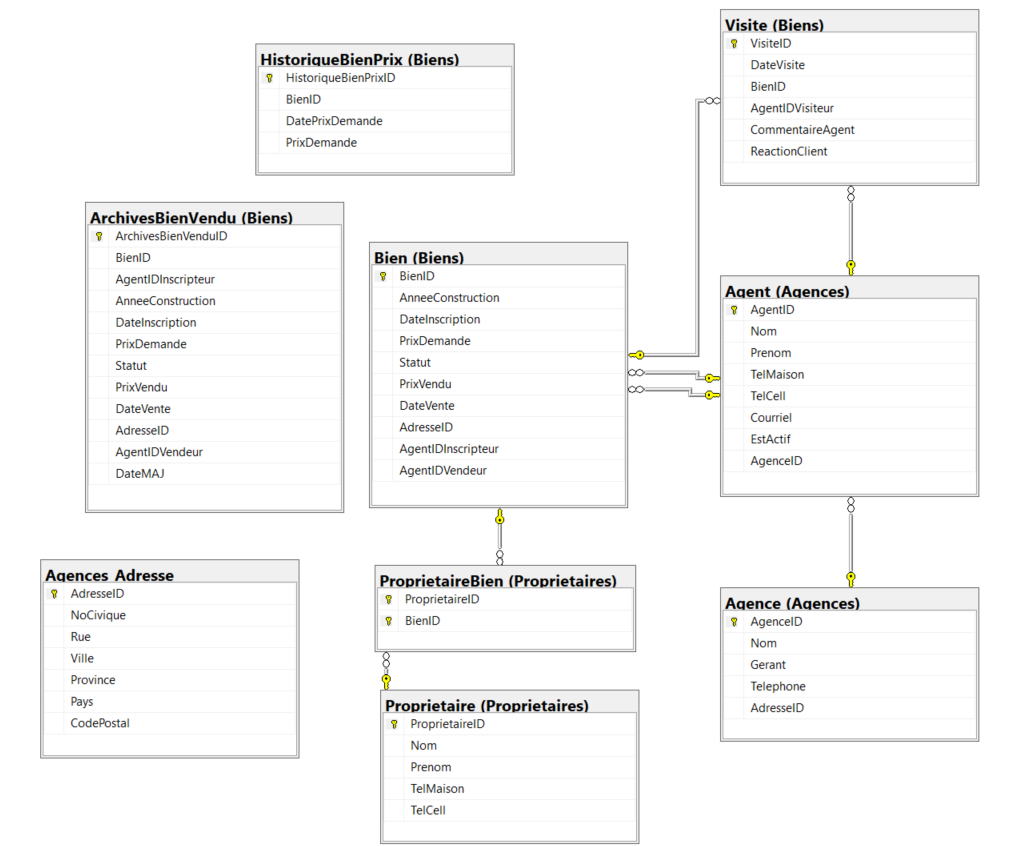
Une fonction et des déclencheurs

**Partie 1 - Un trigger pour deux actions DML 👁**

ATTENTION ! Avant de commencer :

* Exécutez les scripts **420BD5\_Labo05\_create** et **420BD5\_Labo05\_insert** dans SSMS.

Dans cette étape, vous aurez à remplir le fichier **420BD5\_Labo05\_Fonction et Trigger.sql** pour créer un déclencheur qui s’exécutera après l’insertion ou l’update d’un bien.



**PARTIE 1 :**

**Faites un trigger qui se déclenchera quand on fera l'insertion d'un bien** (INSERT) **ou qu'on mettra à jour** (UPDATE) **le champ** **PrixDemande** **de la table** Biens.Bien  **OU le champ** **DateVente** **de la table** Biens.Bien.

Ce trigger s'exécutera **après** l'instruction INSERT ou UPDATE sur la table Biens.Bien.

1. Dans le cas d'un INSERT, on veut insérer dans la table Biens.HistoriqueBienPrix les données BienID, DateInscription et PrixDemande comme premier enregistrement lié à ce bien.
2. Dans le cas d'un UPDATE, il faudra vérifier que l'update a eu lieu sur le champ **PrixDemande** de la table Biens.Bien.

Si tel est le cas, on veut insérer dans la table Biens.HistoriqueBienPrix les données BienID, la date d'aujourd'hui et le nouveau PrixDemande.

Cela nous permettra de garder un historique de fluctuation des prix demandés.

1. Il faudra vérifier que l'update a eu lieu sur le champ **DateVente** de la table Biens.Bien.

Si tel est le cas, on veut ajouter les valeurs de cet enregistrement dans la table Biens.ArchivesBienVendu tout en conservant l'id du bien identique à celui de la table Biens.Bien.

Enlevez ensuite cet enregistrement de la table Biens.Bien.

Supprimez également les données sur les visites sur le bien nouvellement vendu, car nous ne garderons pas les informations sur les visites des biens vendus.

De sorte que dans la table Biens.Bien, il n'y aura, toujours, que les biens non vendus.

Insérez également dans la table Biens.HistoriqueBienPrix les données BienID, la DateVente et le PrixVendu. Ce sera la dernière entrée de notre table Biens.HistoriqueBienPrix pour ce bien.

**PARTIE 2:**

1. **Faites un trigger qui se déclenchera quand on voudra supprimer un agent de la table dbo.Agent parce qu'il quitte l'AGENCE.**

Ce trigger s'exécutera **à la place** de l'instruction DELETE.

Il vérifiera si cet agent est présentement **l'agentIDVendeur** de biens non encore vendus.

Si c'est le cas, il répartira les biens de l'agent qui quitte l'agence pour les donner à l'agent le plus ancien (celui dont l'AgentID est le plus bas) qui a le moins de biens à vendre présentement.

Enfin, il mettra le champ EstActif de la table Agences.Agent à 0 pour l'agent qui quitte l'agence.

CONSEIL : Faites une fonction pour retourner l’id de l’agent le plus ancien ayant le moins de biens à vendre pour le moment, dont l’id est différent de l’id de l’agent qui veut quitter l’agence.

VOTRE TRIGGER utilisera cette fonction.

Faites vos tests que l’agentID 1 quitte l’agence.

Normalement, si votre trigger est bien fait, vous devriez avoir les biens de **l'agentIDVendeur** 1 transférés à l'agent dont l'Id est 5 et ce dernier devrait se retrouver avec 4 biens à gérer:

.