3.3 - Déclencheur standard

POUR CONSULTER LES TRIGGERS :

SELECT name, parent_id, type_desc

FROM sys.triggers;

POUR ON/OFF UN TRIGGER :

-- Désactiver le trigger

DISABLE TRIGGER T_0_nomdutrigger ON nomdelaTABLE;

-- Réactiver le trigger

ENABLE TRIGGER T_0_nomdutrigger ON nomdelaTABLE;

POUR VOIR LESQUEL SONT ACTIVER ET DESACTIVER :

(si = 1 alors le trigger désactiver)

SELECT tr.name AS TriggerNom, tr.is_disabled AS estDésactiver

FROM sys.triggers tr

JOIN sys.objects o ON tr.object_id = o.object_id

WHERE o.type = 'TR'

ORDER BY TriggerNom;

3.3.1 - Gérer la casse d'un texte

- Transformer en majuscule les noms des étudiants (insert, update).

```
CREATE TRIGGER T_01_NomMaj
   ON etudiant
   AFTER INSERT,UPDATE
AS
BEGIN
        UPDATE
                etudiant
        SET
                nom_etu = UPPER(inserted.nom_etu)
        FROM
                etudiant, inserted
        WHERE
                etudiant.num_etu=inserted.num_etu
END
```

Explication : Dès qu'un nouvel étudiant est créé ou mis à jour, son nom (nom_etu) est converti en majuscules.

Pour tester : Rentrer la requête, l'exécuter, actualiser puis créer nouveau étudiant.

Exercice : T_02_NomMaj

Noms des étudiants en majuscule (insert, update).

Première lettre du prénom en majuscule et le reste du prénom en minuscule.

-- il faut penser a désactiver le trigger T_01 -> DISABLE TRIGGER T_01_NomMaj ON Etudiant;

-- Et les autres qui peuvent potentiellement gener

```
CREATE TRIGGER T_02_NomMaj
ON Etudiant
AFTER INSERT, UPDATE
AS
BEGIN
        UPDATE Etudiant
        SET e.nom_etu = UPPER(e.nom_etu), e.prenom_etu = UPPER(LEFT(e.prenom_etu, 1)) +
        LOWER(SUBSTRING(e.prenom_etu, 2, LEN(e.prenom_etu)))
        FROM Etudiant e
        INNER JOIN inserted ON e.num_etu = inserted.num_etu;
END
```

3.3.2 - Archiver automatiquement ;

Exercice : T_03_ HistoLocation

- Copier une ligne supprimée de la table « posséder » dans la table « histoPos ».

```
CREATE TRIGGER T_03_HistoPos
ON posseder
AFTER DELETE
AS
BEGIN
        INSERT INTO histo (date_his, num_etu, num_voi, date_pos, prix_pos)
        SELECT
                getdate(), num_etu, num_voi, date_pos, prix_pos
        FROM
                deleted
END
GO
```

Explication : Dès qu'une donnée est supprimée de la table posséder, elle est copiée dans la table histo.

Pour tester :

- Créer une nouvelle donnée dans la table posséder avec la requête suivante :
  INSERT INTO posseder (num_etu, num_voi, date_pos, prix_pos) VALUES (12, 6, '2024-10-08', 15000);
- Supprimer une ligne dans la table posséder avec la requête suivante :
  DELETE FROM posseder WHERE num_etu = 12 AND num_voi =6;
- Vérifier si les données ont été copiées dans la table histo :
  SELECT * FROM histo

3.3.3 - Modifier une propriété (N1) ;

Exercice : T_04_MajMontantEtudiant

- Mettre à jour la propriété « Montant_total_etu » de la table « etudiant » lors de l'enregistrement d'une possession.

```
CREATE TRIGGER T_04_MajMontantEtudiant
ON Posseder
FOR INSERT, UPDATE AS
BEGIN
	UPDATE
		Etudiant
	SET
		montant_total_etu= montant_total_etu + prix_pos
	FROM
		inserted


	WHERE
			etudiant.num_etu = inserted.num_etu
END
```

Explication : Dès qu'un nouvel étudiant num_etu=14 est créé avec un montant_total_etu = 100 et que dans posséder on reprend ce num_etu=14 avec prix_pos = 50, alors les 2 s'additionne et devienne montant_total_etu = 150

Pour tester :

- Désactiver le trigger 5 -> DISABLE TRIGGER T_05_MajMontantEtudiant2 ON Posseder;
  Et les autres qui peuvent potentiellement gêner
- Créer une nouvelle entrée dans etudiant :
  INSERT INTO etudiant (num_etu, nom_etu, prenom_etu, adr_ville_etu, montant_total_etu) VALUES (14, 'coucou', 'coucou', 'cc', 100);
- Créer une nouvelle entrée dans posséder
  INSERT INTO posseder (num_etu, num_voi, date_pos, prix_pos) VALUES (14, 6, '2024-10-08', 50);
- Vérifier que le montant total de l'étudiant est bien mis à jour :
  SELECT * FROM Etudiant WHERE num_etu=14

3.3.4 - Modifier une propriété (N2) ;

Exercice : T_05_MajMontantEtudiant2

Un déclencheur (INSERT) assure la modification de la propriété « montant_total_etu » de la table « etudiant » en fonction du prix récupéré depuis la table voiture (prix_voi).

```
CREATE TRIGGER [dbo].[T_05_MajMontantEtudiant2]
ON [dbo].[posseder]
FOR INSERT, UPDATE AS
BEGIN
	DECLARE @Prix INT
	SELECT
		@Prix = Prix_Voi
	FROM
		voiture, inserted
	WHERE
		Voiture.Num_Voi = inserted.Num_Voi

	UPDATE
		Etudiant
	SET
		Montant_Total_etu= Montant_Total_etu+Prix_pos
	FROM
		inserted
	WHERE
		etudiant.Num_etu=inserted.Num_etu
END
```

Explication : Un déclencheur (INSERT) met à jour la propriété montant_total_etu de la table etudiant en fonction du prix récupéré depuis la table voiture (colonne prix_voi)

Pour tester :

- Désactiver le trigger 4 -> DISABLE TRIGGER T_04_MajMontantEtudiant ON Posseder;
  Et les autres qui peuvent potentiellement gener
- Créer une nouvelle entrée dans posséder pour l'étudiant et la voiture :
  INSERT INTO posseder (num_etu, num_voi, date_pos, prix_pos) VALUES (21, 50, '2020-1-1', 50;
- Vérifier si la maj a bien eu lieu dans la table etudiant en fonction du prix de la voiture :
  SELECT * FROM Etudiant WHERE num_etu = 21;

3.3.5 - Modifier une propriété (N3) ;

Exercice : T_06_MontantEtudiant2

Le prix cumulé à la propriété « montant_total_etu » dépend de la puissance de la voiture. Si celle-ci est inférieure à 6 cv, le prix correspond à la valeur du prix dans la table voiture, sinon la valeur du prix est promotionnel.

```
CREATE TRIGGER T_06_MontantEtudiant2
ON Posseder
FOR INSERT, UPDATE
AS
BEGIN
  DECLARE @Prix INT, @PrixPromo INT, @Puissance INT;

  -- Récupération du prix, du prix promo et de la puissance de la voiture depuis la table Voiture
  SELECT @Prix = Prix_Voi, @PrixPromo = Prix_promo_voi, @Puissance = Puissance_Voi
  FROM Voiture
  INNER JOIN inserted ON Voiture.Num_Voi = inserted.Num_Voi;

  -- Si la puissance de la voiture est inférieure à 6, ajouter le prix normal
  IF @Puissance < 6
  BEGIN
    UPDATE Etudiant
    SET Montant_Total_etu = Montant_Total_etu + @Prix
    FROM Etudiant
    INNER JOIN inserted ON Etudiant.Num_etu = inserted.Num_etu;
  END
  -- Si la puissance de la voiture est égale ou supérieure à 6, ajouter le prix promotionnel
  ELSE
  BEGIN
    UPDATE Etudiant
    SET Montant_Total_etu = Montant_Total_etu + @PrixPromo
    FROM Etudiant
    INNER JOIN inserted ON Etudiant.Num_etu = inserted.Num_etu;
  END
END
```

Explication : Le trigger met à jour la propriété montant_total_etu dans la table Etudiant en fonction de la puissance de la voiture.

- Si la puissance de la voiture est inférieure à 6 chevaux, le prix de la voiture est utilisé tel quel.
- SI la puissance >= 6 chevaux, un prix promotionnel est appliqué

Pour tester :

- Désactiver les trigger qui peuvent potentiellement gêner
  Rappel : DISABLE TRIGGER T_0_nomdutrigger ON nomdelaTABLE;
- Créer une voiture dans la table Voiture :
  INSERT INTO Voiture (Num_Voi, marque_voi, Puissance_Voi, Prix_voi,Prix_promo_voi, code_cat) VALUES (123,'fff',9, 25000, 900, 3);
- Créer une entrée dans la table Posseder :
  INSERT INTO Posseder (Num_etu, Num_Voi, Date_pos, Prix_pos) VALUES (29, 123, '2024-11-27', 0);
- Vérifier:
  SELECT Montant_Total_etu FROM Etudiant WHERE Num_etu = 29;

Résultat :

	Montant_Total_etu
1	900

3.4 DECLENCHEUR SUR UNE VUE

POUR CONSULTER LES VUES :

SELECT name,create_date,modify_date

FROM sys.views;

3.4.2 - Création de la vue « VoitureCategorie »

- Création 1

```
CREATE VIEW V_01_VoitureCategorie2 AS
SELECT
        Voiture.num_voi, Voiture.Marque_voi,
        categorie.code_cat, categorie.libelle_cat
FROM
        voiture INNER JOIN categorie ON voiture.Code_cat = categorie.Code_cat
```

- Création 2

  Cliquer sur le bouton droit de la souris en se positionnant auparavant sur la vue de la table.
  Choisir « nouvelle vue » et sélectionner les propriétés à visualiser.

- Insertion de données dans la vue VoitureCategorie

  INSERT INTO V_01_VoitureCategorie
  (num_voi, Marque_voi, code_cat, libelle_cat)
  VALUES (50, 'renault', 6, 'collection2')

- Quel est le résultat obtenu ?

Explication :

Cette vue affiche les informations de voiture et leur catégorie associée.

ERREUR : La vue ou la fonction 'V_01_VoitureCategorie2' ne peut pas être mise à jour car la modification porte sur plusieurs tables de base.

Cela génère une erreur car SQL Server ne sait pas comment insérer simultanément dans ces deux tables.

3.4.3 - Amélioration du déclencheur sur la vue « VoitureCategorie »

Exercice : T_08_InsVoitureCategorie

- Création du trigger ci-contre pour éviter l'erreur précédente.

```
CREATE TRIGGER [dbo].[T_08_InsVoitureCategorie] ON [dbo].[V_01_VoitureCategorie]
INSTEAD OF INSERT AS
BEGIN
	INSERT INTO categorie (code_cat, libelle_cat)
	SELECT code_cat, libelle_cat FROM inserted
	INSERT INTO voiture (Num_voi, marque_voi, code_cat)
	SELECT Num_voi, marque_voi, code_cat FROM inserted
END
```

- Insertion de données dans la vue V_01_VoitureCategorie

```
INSERT INTO V_01_VoitureCategorie
(num_voi, marque_voi, code_cat, libelle_cat)
VALUES (50, 'renault', 6, 'collection2')
```

- Quel est le résultat obtenu ?

Explication :

- Cette vue affiche les informations de voiture et leur catégorie associée.

Pour tester :

- INSERT INTO V_01_VoitureCategorie (num_voi, marque_voi, code_cat, libelle_cat) VALUES (500, 'renault', 6, 'collection2')

Résultat :

- SELECT * FROM V_01_VoitureCategorie2 WHERE num_voi = 500;

	num_voi	Marque_voi	code_cat	libelle_cat
1	500	renault	68	collection2

-

- La voiture ‘500’ et sa catégorie associée sont insérées correctement dans leurs tables respectives.

Exercice : T_09_MajLocCli

- Vérifier l'existence du client en modifiant le trigger précédent :
  - La catégorie existe : récupération de son numéro et insérer les informations dans la table « voiture » ;
  - Sinon, insérer les informations dans les tables « catégorie » et « voiture ».
  - Insérer un client existant ;
    INSERT INTO V_01_VoitureCategorie
    (Num_voi, marque_voi, code_cat, libelle_cat)
    VALUES (10, 'Renault', 10, collection2)
  - Insérer un nouveau client ;
    INSERT INTO V_01_VoitureCategorie
    (Num_voi, marque_voi, code_cat, libelle_cat)
    VALUES (11, 'Renault', 10, collection2)

On doit supprimer l'ancien déclencheur ou le modifier :

```
DROP TRIGGER [dbo].[T_08_InsVoitureCategorie];
```

Puis créer le trigger T_09 :

```
CREATE TRIGGER [dbo].[T_09_MajLocCli]
ON [dbo].[V_01_VoitureCategorie2]
INSTEAD OF INSERT
AS
BEGIN

    IF EXISTS (SELECT 1 FROM categorie WHERE code_cat = (SELECT code_cat FROM inserted))
    BEGIN

        INSERT INTO voiture (Num_voi, marque_voi, code_cat)
        SELECT Num_voi, marque_voi, code_cat
        FROM inserted;
    END
    ELSE
    BEGIN

        INSERT INTO categorie (code_cat, libelle_cat)
        SELECT code_cat, libelle_cat FROM inserted;

        INSERT INTO voiture (Num_voi, marque_voi, code_cat)
        SELECT Num_voi, marque_voi, code_cat FROM inserted;
    END
END;
```

Pour tester :

Si le code_cat existe déjà :

INSERT INTO V_01_VoitureCategorie2 (Num_voi, marque_voi, code_cat, libelle_cat)

VALUES (1223, 'fff', 9, 'collection2')

- Insertion dans la table voiture uniquement

➔

	num_voi	marque_voi	puissance_voi	prix_voi	prix_promo_voi	code_cat
1	1223	fff	NULL	NULL	NULL	9

Sinon :

INSERT INTO V_01_VoitureCategorie2 (Num_voi, marque_voi, code_cat, libelle_cat)

VALUES (1239, 'fff', 33, 'collection2')

- Insertion dans la table voiture et categorie :

➔

	num_voi	marque_voi	puissance_voi	prix_voi	prix_promo_voi	code_cat
1	1239	fff	NULL	NULL	NULL	33

➔

	code_cat	libelle_cat
1	33	collection2

3.4.4 - Modification d'informations dans la vue V_01_VoitureCategorie

Exercice : T_10_MajLocCli

- Définir un trigger de modification sur la vue « V_01_VoitureCategorie », exécuté lors de la modification d'un enregistrement dans la vue ;
  - Récupérer les valeurs des données dans la table « inserted » dans des variables ;
  - Modifier les lignes concernées ;

```
CREATE TRIGGER [dbo].[T_10_MajLocCli]
ON [dbo].[V_01_VoitureCategorie2]
INSTEAD OF UPDATE
AS
BEGIN
    -- Récupérer les anciennes et nouvelles valeurs des données modifiées
    DECLARE @OldCodeCat INT, @NewCodeCat INT, @OldLibelleCat VARCHAR(255), @NewLibelleCat VARCHAR(255);
    DECLARE @OldNumVoi INT, @NewNumVoi INT, @OldMarqueVoi VARCHAR(255), @NewMarqueVoi VARCHAR(255);

    -- Anciennes valeurs
    SELECT @OldCodeCat = code_cat, @OldLibelleCat = libelle_cat,
           @OldNumVoi = Num_voi, @OldMarqueVoi = marque_voi
    FROM deleted;

    -- Nouvelles valeurs
    SELECT @NewCodeCat = code_cat, @NewLibelleCat = libelle_cat,
           @NewNumVoi = Num_voi, @NewMarqueVoi = marque_voi
    FROM inserted;

    -- Mise à jour dans la table categorie si le code_cat a changé
    IF @OldCodeCat <> @NewCodeCat
    BEGIN
        UPDATE categorie
        SET code_cat = @NewCodeCat, libelle_cat = @NewLibelleCat
        WHERE code_cat = @OldCodeCat;
    END
    ELSE
    BEGIN
        -- Si le code_cat n'a pas changé, mettre à jour uniquement le libelle_cat
        UPDATE categorie
        SET libelle_cat = @NewLibelleCat
        WHERE code_cat = @OldCodeCat;
    END

    -- Maj dans la table voiture
    UPDATE voiture
    SET Num_voi = @NewNumVoi, marque_voi = @NewMarqueVoi, code_cat = @NewCodeCat
    WHERE Num_voi = @OldNumVoi;
END;
```

Pour tester : (Présentation de chaque tests)

TEST 1 :

UPDATE V_01_VoitureCategorie
SET marque_voi = 'Peugeot'
WHERE Num_voi = 26;

La voiture 26 a changé de marque :

	num_voi	marque_voi
1	26	Peugeot

TEST 2 :

UPDATE V_01_VoitureCategorie2
SET code_cat = 700, libelle_cat = 'Nouvelle catégorie'
WHERE Num_voi = 26;

La voiture 26 a change de code_cat et libelle_cat et création d'une nouvelle categorie :

	num_voi	code_cat
1	26	700

	code_cat	libelle_cat
1	700	Nouvelle catégorie

3.7 PROCEDURE STOCKEE

POUR CONSULTER LES PS :

SELECT name,create_date,modify_date

FROM sys.procedures;

SANS PARAMETRE

3.7.4 - Procédure stockée sans paramètre

- Afficher les étudiants qui possèdent une voiture.

```
CREATE PROCEDURE PS_01_EtuPosVoi
AS
BEGIN
        SELECT etudiant.num_etu, nom_etu, prenom_etu, adr_ville_etu, montant_total_etu
        FROM etudiant, posseder
        WHERE etudiant.num_etu = posseder.num_etu
END
GO
```

- Création d'une nouvelle requête ;

```
EXEC PS_01_EtuPosVoi
```

Explication : Créer une procedure stocké pour afficher le numero, le nom, prenom, ville, montant total etu des étudiants qui possèdent une voiture

Pour tester :
EXEC PS_01_EtuPosVoi ;

Résultat : (extrait)

	num_etu	nom_etu	prenom_etu	adr_ville_etu	montant_total_etu
1	2	Dubois	Lucas	Nimes	0
2	2	Dubois	Lucas	Nimes	0
3	5	Dupont	Lisa	Paris	0
4	5	Dupont	Lisa	Paris	0
5	5	Dupont	Lisa	Paris	0
6	12	Martin	Pierre	Nimes	0
7	12	Martin	Pierre	Nimes	0

AVEC PARAMETRE

3.7.5 - Procédure stockée avec paramètre

3.7.5.1 - Notions

- 1024 paramètres possible ;
- Faire précéder l'identificateur du caractère @

3.7.5.2 - Liste des voitures (numéro, marque, puissance_voi, prix_voi) possédées par un étudiant donné (nom_etu).

- Création ;

```
CREATE PROCEDURE PS_02_VoiEtu @NomEtudiant char(50)
AS
BEGIN
        SELECT
                Posseder.num_voi, marque_voi, puissance_voi, prix_voi
        FROM Voiture, Posséder, Etudiant
        WHERE
                Voiture.Num_voi = Posseder.Num_voi
        AND   etudiant.Num_etu = Posseder.Num_etu
        AND   etudiant.Nom_etu = @NomEtudiant
END
GO
```

- Exécution ;

```
EXEC PS_02_VoiEtu 'dupont'
```

Explication :

La procédure prend un paramètre @NomEtudiant et fais jointure avec les table voiture, posseder, et etudiant.

Elle retourne les informations sur les voitures possédées par l'étudiant @NomEtudiant.

Pour tester :
EXEC PS_02_VoiEtu 'dupont' ;

Résultat : (extrait)

	num_voi	marque_voi	puissance_voi	prix_voi
1	1	Peugeot	4	15000.00
2	4	Renault	7	17000.00
3	6	Renault	6	19000.00

3.7.5.3 - Prix d'une voiture (prix_promo_voi, prix_voi) possédée en fonction de 2 dates données en paramètres

- Création ;

```
CREATE PROCEDURE PS_04_PrixVoiture2Dates
                    @DateDeb date,
                    @DateFin date,
                    @PrixVoiture money OUTPUT,
                    @PrixPromoVoiture money OUTPUT
AS
BEGIN
    SELECT @PrixVoiture = Prix_voi, @PrixPromoVoiture = Prix_promo_voi
    FROM voiture, posseder
    WHERE voiture.num_voi = posseder.num_voi
    AND date_pos >= @DateDeb
    AND date_pos <= @DateFin
END
GO
```

Explication :

Cette procédure prend deux paramètres d'entrée @DateDeb et @DateFin.

Elle retourne deux valeurs en sortie : @PrixVoiture et @PrixPromoVoiture.

Elle sélectionne les prix d'une voiture possédé entre les deux dates spécifiées.

Pour tester : (On peux combiner des chaînes et des variables pour obtenir un affichage plus lisible avec la fonction CONCAT)

- Exécution ;
  Attention, évitez d'avoir des valeurs nulles dans les propriétés à afficher.

```
DECLARE @Prix1 float
 DECLARE @Prix2 float
 EXEC PS_04_PrixVoiture2Dates
 '2018-01-01',
 '2020-01-31',
 @PrixVoiture=@Prix1 OUTPUT,
 @PrixPromoVoiture=@Prix2 OUTPUT
PRINT CONCAT('Prix de la voiture : ', @Prix1, ' | Prix promo : ', @Prix2)
```

Résultat : (extrait)

```
Messages
Prix de la voiture : 1000 | Prix promo : 900
```

3.7.5.4 - Gestion des erreurs : try...catch

- Notions ;

  Intercepter une erreur ;
- Création ;

```
CREATE PROCEDURE PS_05_EnregVoiCat
                @NumVoi int,
                @MarqueVoi varchar(30),
                @PrixVoi money,
                @PrixPromoVoi money,
                @CodeCat int,
                @LibelleCat varchar(30)
 AS
 BEGIN TRY
                -- insertion dans les tables
                INSERT INTO Voiture (Num_voi, Marque_Voi, Prix_voi, Prix_Promo_voi, Code_cat)
                VALUES (@NumVoi, @MarqueVoi, @PrixVoi, @PrixPromoVoi, @CodeCat)

                INSERT INTO Categorie (Code_Cat, Libelle_cat)
                VALUES (@CodeCat, @LibelleCat)
 END TRY
 BEGIN CATCH
            SELECT
                 ERROR_NUMBER() AS NumeroErreur,
                 ERROR_SEVERITY() AS SeverityErreur,
                 ERROR_STATE() AS NumeroErreur,
                 ERROR_PROCEDURE() AS ProcedureErreur,
                 ERROR_LINE() AS LigneErreur,
                 ERROR_MESSAGE() AS MessageErreur
 END CATCH
```

Explication :

BEGIN TRY tente d'exécuter les instructions d'insertion.

BEGIN CATCH intercepte les erreurs et affiche des informations sur celles-ci.

Pour tester :

- EXEC PS_05_EnregVoiCat null, "Renault", 100, 110, 1, "categ2"

	NumeroErreur	SeverityErreur	NumeroErreur	ProcedureErreur	LigneErreur	MessageErreur
1	515	16	2	PS_05_EnregVoiCat	11	Impossible d'insérer la valeur NULL dans la colonne 'num_voi', table 'Faculté.dbo.voiture'.

- EXEC PS_05_EnregVoiCat 100, "Renault", 100, 110, 1, "categ2"

	NumeroErreur	SeverityErreur	NumeroErreur	ProcedureErreur	LigneErreur	MessageErreur
1	2627	14	1	PS_05_EnregVoiCat	13	Violation de la contrainte PRIMARY KEY « PK_categorie ». Impossible d'insérer une clé en do

TRIGGER – VIEW – PS

```
CREATE PROCEDURE PS_06_NbEtudiantVille
 @VilleEtu varchar(50) OUTPUT,
 @NbEtu int OUTPUT
 AS
 BEGIN
 SELECT @NbEtu = COUNT(*)
 FROM etudiant
 WHERE adr_ville_etu = @VilleEtu
 END
 GO
```

Explication :

Affiche le nombre de clients dune ville

Pour tester :

```
DECLARE @NbEtudiant int
EXEC PS_06_NbEtudiantVille
    @VilleEtu = 'Nimes',
    @NbEtu = @NbEtudiant OUTPUT
SELECT @NbEtudiant AS NombreDeEtudiantParVille
```

Resultat :

	NombreDeEtudiantParVille
1	5

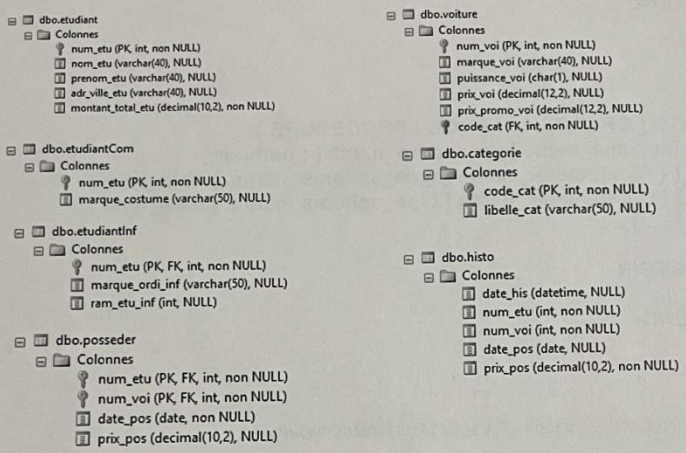
Nom : Prénom Durée : 1h00

BTS SIO Auteur : Eric Willems 26/11/2024

BREVET DE TECHNICIEN SUPERIEUR
SERVICES INFORMATIQUES AUX ORGANISATIONS
SISR- SLAM

Téléphone portable et documents sont interdits

Sujet



1 - Réaliser le MCD du schéma relationnel ci-dessus.

2 - Réaliser un trigger.
Copier une ligne supprimée de la table « posseder » dans la table « histo ».

Pour les PS, Indiquer votre jeu de données pour vérifier tous les cas d'exécution.

3 – Réaliser une PS
Soit X, Z correspondants à des paramètres de la procédure stockée.
Afficher les prénoms des étudiants dont le nom commence par X et qui ne possèdent pas de voiture de marque Z.

4 – Réaliser une PS
Soit NB, M, A correspondants à des paramètres de la procédure stockée.
Afficher les noms et prénoms des étudiants qui possèdent plus de NB voiture pendant le mois M de l'année A

1/ 2

2- Réaliser un trigger.

Copier une ligne supprimée de la table <<< posseder >> dans la table << histo >>.

Déjà fais au 3.3.2

TRIGGER – VIEW – PS

3-Réaliser une PS

Soit X, Z correspondants à des paramètres de la procédure stockée.

Afficher les prénoms des étudiants dont le nom commence par X et qui ne possèdent pas de voiture de marque Z.

```
CREATE PROCEDURE PS_ExerciceEval_3_Parametre
    @X CHAR(1),      -- premier caractère du nom
    @Z VARCHAR(50)   -- marque de la voiture
AS
BEGIN
    SELECT e.prenom_etu
    FROM Etudiant e
    WHERE e.nom_etu LIKE @X + '%'
    AND e.num_etu NOT IN (
        SELECT p.num_etu
        FROM Posseder p
        JOIN Voiture v ON p.num_voi = v.num_voi
        WHERE v.marque_voi = @Z
    );
END;
```

Explication : La procédure doit retourner les prénoms des étudiants dont le nom commence par "X" et qui ne possèdent pas de voiture de marque "Z"

Pour tester :

EXEC PS_03_EtuSansVoitureMarqueZ @X = 'D', @Z = 'Renault';

Données :

Table voiture :

num_voi	marque_vo
1	Renault
4	Renault

Table etudiant :

num_etu	nom_etu	prenom_et
1	DULONG	Jean
2	DUBOIS	Lucas
5	Dupont	Lisa

Table posséder :

num_etu	num_voi
1	1
2	2
2	4
2	8
5	1
5	4

Affichage Résultat : - >

	prenom_etu
1	Jean

TRIGGER – VIEW – PS

4- Réaliser une PS

Soit NB, M, A correspondants à des paramètres de la procédure stockée.

Afficher les noms et prénoms des étudiants qui possèdent plus de NB voiture pendant le mois M de l'année A

```
CREATE PROCEDURE PS_ExerciceEval_4_Parametre
    @NB INT,       -- nb voiture
    @M INT,        -- mois
    @A INT         -- année
AS
BEGIN

    SELECT Etudiant.nom_etu, Etudiant.prenom_etu
    FROM Etudiant
    JOIN Posseder ON Etudiant.num_etu = Posseder.num_etu
    JOIN Voiture ON Posseder.num_voi = Voiture.num_voi
    WHERE MONTH(Posseder.date_pos) = @M
    AND YEAR(Posseder.date_pos) = @A
    GROUP BY Etudiant.nom_etu, Etudiant.prenom_etu
    HAVING COUNT(Posseder.num_voi) > @NB
END
```

Explication : La procédure doit retourner les noms et prénoms des étudiants qui possèdent plus de **NB** voitures pendant le mois **M** de l'année **A**.

Pour tester :

EXEC PS_ExerciceEval_4_Parametre @NB = 1, @M = 1, @A = 2020;

Données :

Table etudiant :

num_etu	nom_etu	prenom_etu
1	DULONG	Jean
2	DUBOIS	Lucas

Table posséder :

num_etu	num_voi	date_pos
1	1	2020-02-02
2	2	2020-02-02
2	4	2020-01-12
2	8	2020-01-12

Affichage Résultat : - >

	nom_etu	prenom_etu
1	DUBOIS	Lucas