



Rapport des travaux : TP T-SQL & Projet BDD

Sous l'encadrement de Pr. Mostafa Ezziyyani

Préparé par :

El ghazi Loubna

Zaoui Hanane

2023/2024

SOMMAIRE

1. INTRODUCTION.....	3
2. REALISATION DU TP T-SQL	4
3. REALISATION DU PROJET BDD.....	22
3.1. But de ce projet.....	22
3.2. Outils utilisés	22
3.3. Solution.....	23
3.4 Démonstration	24
4. CONCLUSION.....	39
5. WEBOGRAPHIE.....	40

1. INTRODUCTION

Ce rapport présente le fruit de notre travail dans le cadre du module de Base de Données Avancées.

L'objectif principal de ce travail était de consolider nos connaissances dans les fondamentaux des bases de données et d'explorer des concepts avancés tels que le SQL dynamique, les procédures/fonctions stockées, les curseurs et l'indexation arbres B+.

Nous avons entrepris la réalisation d'une application qui met en œuvre ces différents concepts, dans le but de les comprendre plus en profondeur et de les appliquer de manière pratique. Ce projet nous a permis de renforcer notre compréhension des bases de données relationnelles et d'acquérir des compétences précieuses dans la manipulation des données et l'optimisation des requêtes.

Nous tenons à exprimer notre profonde gratitude envers notre professeur, **Mostafa Ezziyyani**, pour son encadrement attentif et ses précieux conseils tout au long de ce module. Ses connaissances approfondies et son dévouement ont grandement contribué à notre apprentissage et à notre réussite dans ce domaine complexe.

Dans les sections suivantes de ce rapport, nous présenterons en détail le projet que nous avons réalisé, en mettant en évidence les différentes fonctionnalités implémentées et les solutions apportées.

2. REALISATION DU TP T-SQL

1. Ecrire une procédure stockée qui permet de trouver tous les entiers inférieurs à un nombre donné dont la somme de ces chiffres égale à 6. En stockant les informations suivantes dans une table temporaire:

- a. le nombre de chiffres paires
- b. le nombre de chiffres impaire

Réponse :

```
CREATE PROCEDURE TrouverEntiersInf6 (@Nombre INT)
AS
BEGIN
    CREATE TABLE #TMP (
        Nombre INT,
        ChiffresPairs INT,
        ChiffresImpairs INT
    )

    DECLARE @Entier INT
    DECLARE @Chiffre INT
    DECLARE @SommeChiffres INT
    DECLARE @ChiffresPairs INT
    DECLARE @ChiffresImpairs INT

    SET @Entier = 0
    WHILE @Entier < @Nombre
    BEGIN
        SET @Entier = @Entier + 1
        SET @SommeChiffres = 0
        SET @ChiffresPairs = 0
        SET @ChiffresImpairs = 0

        DECLARE @NumString VARCHAR(10)
        SET @NumString = CAST(@Entier AS VARCHAR(10))

        DECLARE @Index INT
        SET @Index = 1

        WHILE @Index <= LEN(@NumString)
        BEGIN
            SET @Chiffre = CAST(SUBSTRING(@NumString, @Index, 1) AS INT)
            SET @SommeChiffres = @SommeChiffres + @Chiffre
```

```

        IF @Chiffre % 2 = 0
            SET @ChiffresPairs = @ChiffresPairs + 1
        ELSE
            SET @ChiffresImpairs = @ChiffresImpairs + 1

        SET @Index = @Index + 1
    END

    IF @SommeChiffres = 6
    BEGIN
        INSERT INTO #TMP (Nombre, ChiffresPairs, ChiffresImpairs)
        VALUES (@Entier, @ChiffresPairs, @ChiffresImpairs)
    END

END

SELECT * FROM #TMP
DROP TABLE #TMP
END

```

2. Ecrire une fonction stockée qui permet de calculer le code binaire d'un entier

(la valeur de retour de la fonction est une chaîne de caractères).

```

CREATE FUNCTION CalculerCodeBinaire2(@Entier INT)
RETURNS VARCHAR(32)
AS
BEGIN
    RETURN CONVERT(VARCHAR(32), @Entier, 2)
END

```

3. Ecrire une fonction stockée qui permet de tester si une chaîne de caractères est un palindrome (Exemple : TOTOT et TOUSUOT sont deux palindromes).

```

CREATE FUNCTION Palindrome (@Chaine VARCHAR(100))
RETURNS BIT
AS
BEGIN
    DECLARE @ReverseChaine VARCHAR(100)
    SET @ReverseChaine = REVERSE(@Chaine)

    IF @Chaine = @ReverseChaine
        RETURN 1
    ELSE
        RETURN 0
    RETURN 0
END

```

4. Ecrire une fonction stockée qui permet de compter le nombre des mots dans une chaîne de caractère passer en paramètre.

```
SQLQuery4.sql - D:\686R\LOUBNA (55)* - X SQLQuery3tpex3.s...686R\LOUBNA (52)) SQLQuery2tpex1.s...
CREATE FUNCTION CompterMots (@Chaine VARCHAR(100))
RETURNS INT
AS
BEGIN
    DECLARE @NbMots INT
    SET @NbMots = LEN(@Chaine) - LEN(REPLACE(@Chaine, ' ', '')) + 1

    RETURN @NbMots
END
```

5. Ecrire une fonction stockée qui permet de compter le nombre d'occurrences d'une chaîne dans une autre chaîne de caractères.

```
SQLQuery5.sql - D:\686R\LOUBNA (57)* - X SQLQuery4tpex4.s...686R\LOUBNA (55)) SQLQuery3tpex3.s...686R\LOUBNA (52))
CREATE FUNCTION CompterOccurrences(@Chaine VARCHAR(100), @SousChaine VARCHAR(100))
RETURNS INT
AS
BEGIN
    RETURN (LEN(@Chaine) - LEN(REPLACE(@Chaine, @SousChaine, ''))) / LEN(@SousChaine)
END
```

6. Ecrire une fonction stockée qui permet de trouver le plus long mot dans une chaîne de caractères.

```
SQLQuery6.sql - D:\686R\LOUBNA (59)* - X SQLQuery5tpex5.s...686R\LOUBNA (57)) SQLQuery4tpex4.s...686R\LOUBNA (55))
CREATE FUNCTION TrouverPlusLongMot(@Chaine VARCHAR(1000))
RETURNS VARCHAR(100)
AS
BEGIN
    DECLARE @Mots TABLE (Mot VARCHAR(100))
    DECLARE @MotPlusLong VARCHAR(100)

    INSERT INTO @Mots
    SELECT value FROM STRING_SPLIT(@Chaine, ' ')

    SELECT @MotPlusLong = MAX(Mot)
    FROM @Mots
    WHERE LEN(Mot) = (SELECT MAX(LEN(Mot)) FROM @Mots)

    RETURN @MotPlusLong
END
```

7. Ecrire une procédure stockée qui permet d'afficher un nombre des minutes X sous la forme : AA Années MM Mois JJJ Jours HH Heures MM Minutes, sans utiliser les fonctions natives prédéfinies .

```
SQLQuery8.sql - D:\686R\LOUBNA (50)* - X SQLQuery7TPex2.s...686R\LOUBNA (38)) SQLQuery6TPex6.s...686R\LOUBNA (79))
CREATE PROCEDURE AfficherTempsEcoule
    @Minutes INT
AS
BEGIN
    DECLARE @Annees INT, @Mois INT, @Jours INT, @Heures INT

    SET @Annees = @Minutes / (60 * 24 * 365)
    SET @Minutes = @Minutes % (60 * 24 * 365)
    SET @Mois = @Minutes / (60 * 24 * 30)
    SET @Minutes = @Minutes % (60 * 24 * 30)
    SET @Jours = @Minutes / (60 * 24)
    SET @Minutes = @Minutes % (60 * 24)
    SET @Heures = @Minutes / 60
    SET @Minutes = @Minutes % 60

    SELECT CONCAT(@Annees, ' Années ', @Mois, ' Mois ', @Jours, ' Jours ', @Heures, ' Heures ', @Minutes, ' Minutes') AS Temps
END
EXEC AfficherTempsEcoule 15000;
```

8. Ecrire une procédure stockée qui permet la création de la table Vols (Avec la prise en considération toutes les contraintes : d'intégrités structurelles, référentielles, unicité de la clés...)

```
CREATE PROCEDURE CreerTableVols
AS
BEGIN
    CREATE TABLE Vols (
        Num_Vol INT PRIMARY KEY,
        Date_Depart DATE,
        Heure_Depart TIME,
        Ville_Depart VARCHAR(50),
        Ville_Arrivee VARCHAR(50),
        Code_Avion INT,
        Code_Pilote INT,
        Prix_Vol FLOAT,
        FOREIGN KEY (Code_Avion) REFERENCES Avions(Num_Avion),
        FOREIGN KEY (Code_Pilote) REFERENCES Pilotes(Num_Pilote),
        CONSTRAINT CHK_Date_Depart CHECK (Date_Depart >= GETDATE())
    );
    PRINT 'La table Vols a été créée avec succès.'
END
```

9. Ecrire une procédure stockée qui permet d'afficher toutes les réservations non validées à une date donnée.

```
CREATE PROCEDURE AfficherReservationsNonValidees
    @DateDonnee DATE
AS
BEGIN
    SELECT *
    FROM Reservations
    WHERE Date_Validation IS NULL
    AND Date_Reservation = @DateDonnee;
END

EXEC AfficherReservationsNonValidees '2024-03-16';
```

10-11. Ecrire une procédure stockée qui permet d'afficher toutes les informations d'un vol donnée.

```
CREATE PROCEDURE AfficherInformationsVol
    @NumVol INT
AS
BEGIN
    SELECT *
    FROM Vols
    WHERE Num_Vol = @NumVol;
END

EXEC AfficherInformationsVol @NumVol = 101;
```

Num_Vol	Date_Depart	Heure_Depart	Ville_Depart	Ville_Arrivee	Code_Avion	Code_Pilote	Prix
101	2024-04-26	10:00:00.0000000	Casablanca	Marrakech	1	1	101

12. Ecrire une procédure stockée qui permet d'afficher toutes les informations d'une réservation validée (Billet).

```
CREATE PROCEDURE AfficherInfosReservationValidee
AS
BEGIN
    SELECT B.*, R.*
    FROM Billets B
    INNER JOIN Reservations R ON B.Num_Reservation = R.Num_Reservation
    WHERE R.Etat_Reservation = 'Validé';
END;

EXEC AfficherInfosReservationValidee;
```

	Num_Billet	Num_Reservation	Num_Reservation	Date_Reservation	Date_Validation	Etat_Reservation	Code_Agence	Code_
1	201	101	101	2024-03-15	2024-03-16	Validé	1	1

13. Ecrire une procédure stockée qui permet d'afficher le nombre des voyage de chaque avion dans l'ordre décroissant.

```
CREATE PROCEDURE AfficherNombreVoyagesParAvion
AS
BEGIN
    SELECT V.Code_Avion, COUNT(VO.Num_Vol) AS NombreVoyages
    FROM Vols V
    INNER JOIN Voyages VO ON V.Num_Vol = VO.Num_Vol
    GROUP BY V.Code_Avion
    ORDER BY NombreVoyages DESC;
END;
```

14. Ecrire une fonction stockée qui permet de calculer le nombre des voyages d'un passager donnée.

```
CREATE FUNCTION CalculerNombreVoyagesPassager (@CodePassager INT)
RETURNS INT
AS
BEGIN
    DECLARE @NombreVoyages INT;

    SELECT @NombreVoyages = COUNT(Num_Vol)
    FROM Voyages
    WHERE Code_Passager = @CodePassager;

    RETURN @NombreVoyages;
END;
```

15. Ecrire une Fonction stockée qui permet de calculer le prix de revient d'un vol donné.

```
CREATE FUNCTION CalculerPrixRevientVol (@NumVol INT)
RETURNS FLOAT
AS
BEGIN
    DECLARE @PrixRevient FLOAT;

    SELECT @PrixRevient = SUM(Prix_Vol)
    FROM Vols
    WHERE Num_Vol = @NumVol;

    RETURN @PrixRevient;
END;
```


16. Ecrire une procédure stockée qui permet de supprimer toutes les réservations non validées.

```
CREATE PROCEDURE SupprimerReservationsNonValidees
AS
BEGIN
    DELETE FROM Reservations
    WHERE Etat_Reservation <> 'Validé';
END;
EXEC SupprimerReservationsNonValidees;
```

17. Ecrire une procédure stockée qui permet d'insérer un enregistrement dans la table Voyages sous les contraintes suivantes :

- o Teste de l'unicité des enregistrements dans la table Voyages
- o Contrôle si le numéro du billet est correspond bien le passager et le vol.
- o Teste de l'unicité du numéro de la place accordée au passager.

```
CREATE PROCEDURE InsérerVoyage1 (
    @Code_Passager INT,
    @Num_Billet INT,
    @Num_Vol INT,
    @Num_Place INT
)
AS
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Voyages WHERE Code_Passager = @Code_Passager AND Num_Billet = @Num_Billet AND Num_Vol = @Num_Vol)
    BEGIN
        IF EXISTS (SELECT 1 FROM Billets WHERE Num_Billet = @Num_Billet AND Code_Passager = @Code_Passager AND Num_Reservation = @Num_Vol)
        BEGIN
            IF NOT EXISTS (SELECT 1 FROM Voyages WHERE Code_Passager = @Code_Passager AND Num_place = @Num_Place)
            BEGIN
                INSERT INTO Voyages (Code_Passager, Num_Billet, Num_Vol, Num_place)
                VALUES (@Code_Passager, @Num_Billet, @Num_Vol, @Num_Place);
                PRINT 'L'enregistrement a été inséré avec succès dans la table Voyages.';
            END
            ELSE
            BEGIN
                PRINT 'Erreur : Le numéro de place accordée au passager est déjà utilisé.';
            END
        END
        ELSE
        BEGIN
            PRINT 'Erreur : Le numéro du billet ne correspond pas au passager et au vol spécifiés.';
        END
    END
    ELSE
    BEGIN
        PRINT 'Erreur : L'enregistrement existe déjà dans la table Voyages.';
    END
END
```

18. Ecrire une procédure stockée qui permet d'insérer un enregistrement dans la table Ligne_Reservation sous les contraintes suivantes :

- o Teste de l'unicité de la clés de la table Ligne_Reservation
- o contrôle si le numéro d'ordre est sérial pour la nouvelle réservation.
- o contrôle si la ville de départ du vol de la nouvelle réservation coïncide avec la ville d'arrivé du vol de réservation précédente (à l'exception du premier vol).
- o contrôle s'il y a encore une place dans l'avion.

```

CREATE PROCEDURE InsereLigneReservation (
    @Num_Ligne INT,
    @Num_Order INT,
    @Num_Vol INT,
    @Num_Reservation INT
)
AS
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Ligne_Reservation WHERE Num_Ligne = @Num_Ligne)
    BEGIN
        IF @Num_Order = (SELECT ISNULL(MAX(Num_Order), 0) + 1 FROM Ligne_Reservation WHERE Num_Reservation = @Num_Reservation)
        BEGIN
            IF @Num_Order = 1 OR (SELECT Ville_Depart FROM Vols WHERE Num_Vol = @Num_Vol) = (SELECT Ville_Arrivee FROM Vols WHERE Num_Vol = (SELECT Num_Vol FROM Ligne_Reservation WHERE Num_Ligne = @Num_Ligne - 1))
            BEGIN
                IF (SELECT COUNT(*) FROM Voyages WHERE Num_Vol = @Num_Vol) < (SELECT Nbr_Place FROM Avions WHERE Num_Avion = (SELECT Num_Avion FROM Vols WHERE Num_Vol = @Num_Vol))
                BEGIN
                    INSERT INTO Ligne_Reservation (Num_Ligne, Num_Order, Num_Vol, Num_Reservation)
                    VALUES (@Num_Ligne, @Num_Order, @Num_Vol, @Num_Reservation);
                    PRINT 'Enregistrement inséré avec succès.';
                END
                ELSE PRINT 'Erreur : Plus de place disponible dans l''avion pour ce vol.';
            END
            ELSE PRINT 'Erreur : La ville de départ du vol ne coïncide pas avec la ville d''arrivée du vol précédent.';
        END
        ELSE PRINT 'Erreur : Le numéro d''ordre n''est pas sérial pour la nouvelle réservation.';
    END
    ELSE PRINT 'Erreur : L''unicité de la clé de la table Ligne_Reservation est violée.';
END;
EXEC InsereLigneReservation @Num_Ligne = 5, @Num_Order = 2, @Num_Vol = 102, @Num_Reservation = 107;

```

19. Ecrire une procédure stockée qui permet d'ajouter deux colonnes Nbr_Res, et Nbr_Att dans la table Vols pour stocker respectivement le nombre de places réservées et le nombre de places attribuées pour chaque vol, et initialiser les deux colonnes par 0 (Utiliser SQL dynamique avec la commande EXECUTE IMMEDIATE).

```

CREATE PROCEDURE AjouterColonnesNbrResNbrAtt1
AS
BEGIN
    IF NOT EXISTS (
        SELECT 1
        FROM INFORMATION_SCHEMA.COLUMNS
        WHERE TABLE_NAME = 'Vols' AND COLUMN_NAME IN ('Nbr_Res', 'Nbr_Att')
    )
    BEGIN
        ALTER TABLE Vols
        ADD Nbr_Res INT NOT NULL DEFAULT 0,
            Nbr_Att INT NOT NULL DEFAULT 0;

        PRINT 'Colonnes Nbr_Res et Nbr_Att ajoutées avec succès à la table vols.';
    END
    ELSE
    BEGIN
        PRINT 'Les colonnes Nbr_Res et Nbr_Att existent déjà dans la table vols.';
    END
END;
EXEC AjouterColonnesNbrResNbrAtt1;
Select * From Vols;

```

	Num_Vol	Date_Depart	Heure_Depart	Ville_Depart	Ville_Arrivee	Code_Avion	Code_Pilote	Prix_Vol	Nbr_Res	Nbr_Att
1	101	2024-04-26	10:00:00.0000000	Casablanca	Marrakech	1	1	100	0	0
2	102	2024-04-26	11:30:00.0000000	Rabat	Tanger	2	2	120	0	0
3	103	2024-04-03	09:45:00.0000000	Fès	Agadir	1	1	200	0	0
4	104	2024-03-29	08:15:00.0000000	Marrakech	Oujda	2	2	150	0	0

20. Ecrire une procédure stockée qui permet de mettre à jours les deux colonnes Nbr_Res, et Nbr_Att respectivement par le nombre de places réservées et le nombre de places attribuées pour un vol donné.

```

CREATE PROCEDURE MettreAJourNbrPlaces (
    @Num_Vol INT
)
AS
BEGIN
    UPDATE Vois
    SET Nbr_Res = (
        SELECT COUNT(*) FROM Voyages WHERE Num_Vol = @Num_Vol
    ),
        Nbr_Att = (
        SELECT COUNT(*) FROM Voyages WHERE Num_Vol = @Num_Vol AND Num_Place IS NOT NULL
        )
    WHERE Num_Vol = @Num_Vol;
END;
EXEC MettreAJourNbrPlaces @Num_Vol = 101;
SELECT * FROM Vois;

```

Num_Vol	Date_Depart	Heure_Depart	Ville_Depart	Ville_Arribee	Code_Avion	Code_Pilote	Prix_Vol	Nbr_Res	Nbr_Att
101	2024-04-26	10:00:00.0000000	Casablanca	Marrakech	1	1	100	1	1
102	2024-04-26	11:30:00.0000000	Rabat	Tanger	2	2	120	0	0
103	2024-04-03	09:45:00.0000000	Fès	Agadir	1	1	200	0	0
104	2024-03-29	08:15:00.0000000	Marrakech	Oujda	2	2	150	0	0

21. Ecrire une procédure stockée qui permet de calculer le champ « Catégorie » d'un passager donné selon les cas suivants :

- o 'Très Actif' : Pour les passagers qui ont effectués plus de 20 voyages avec un montant de paiement dépasse 200 000 durant l'année en cours.
- o 'Actif' : Pour les passagers qui ont effectués plus de 20 voyages durant l'année en cours.
- o 'Moyen' : Pour les autres passagers

```

CREATE PROCEDURE CalculerCategoriePassager (
    @Code_Passager INT
)
AS
BEGIN
    DECLARE @NombreVoyages INT;
    DECLARE @MontantTotal FLOAT;

    SELECT @NombreVoyages = COUNT(*)
    FROM Voyages
    WHERE Code_Passager = @Code_Passager;

    SELECT @MontantTotal = SUM(Prix_Total)
    FROM Reservations
    WHERE Code_Passager = @Code_Passager;

    IF @NombreVoyages > 20 AND @MontantTotal > 200000
        UPDATE Passagers SET Categorie = 'Très Actif' WHERE Code_Passager = @Code_Passager;
    ELSE IF @NombreVoyages > 20
        UPDATE Passagers SET Categorie = 'Actif' WHERE Code_Passager = @Code_Passager;
    ELSE
        UPDATE Passagers SET Categorie = 'Moyen' WHERE Code_Passager = @Code_Passager;
END;
EXEC CalculerCategoriePassager @Code_Passager = 1;
SELECT * FROM Passagers;

```

Code_Passager	Nom_Passager	Pre_Passager	Num_Passport	Categorie	Num_Tel
1	El Ghazi	Loubna	AB123456	Moyen	212345678
2	Zaoui	Hanane	CD789012	Enfant	69012345
3	Berrada	Mohammed	EF345678	Adulte	356789012

22. Ecrire une procédure stockée qui permet de calculer le nombre des voyages de chaque passager.

```

SQLQuery22.sql - ...:1080K\LOUBINA (88)) - P X SQLQuery20Ex25.sql - not connected SQLQuery19ex24.sql - not connected SQLQuery18ex23.sql - not connected
CREATE PROCEDURE CalculerNombreVoyagesPassager3
AS
BEGIN
SELECT p.Code_Passager, p.Nom_Passager, p.Pre_Passager, COUNT(v.Num_Vol) AS NombreVoyages
FROM Passagers p
LEFT JOIN Voyages v ON p.Code_Passager = v.Code_Passager
GROUP BY p.Code_Passager, p.Nom_Passager, p.Pre_Passager;
END;
EXEC CalculerNombreVoyagesPassager3;

```

23. Ecrire une procédure stockée qui permet de calculer le coût de revient de tous les vols.

```

SQLQuery22.sql - ...:1080K\LOUBINA (88)) - P X SQLQuery20Ex25.sql - not connected SQLQuery19ex24.sql - not connected SQLQuery18ex23.sql - not connected
CREATE PROCEDURE CalculerCoutRevientVols
AS
BEGIN
SELECT v.Num_Vol, v.Prix_Vol, COUNT(DISTINCT r.Num_Reservation) AS NombreReservations,
SUM(v.Prix_Vol) AS CoutTotal, SUM(r.Prix_Total) AS Recettes,
SUM(v.Prix_Vol) - SUM(r.Prix_Total) AS CoutRevient
FROM Vols v
LEFT JOIN Ligne_Reservation lr ON v.Num_Vol = lr.Num_Vol
LEFT JOIN Reservations r ON lr.Num_Reservation = r.Num_Reservation
GROUP BY v.Num_Vol, v.Prix_Vol;
END;
EXEC CalculerCoutRevientVols;

```

24. Même question (14), pour tous les passagers.

```

Query19.sql - ...:1080K\LOUBINA (81)) - P X SQLQuery18ex23.s...:1080K\LOUBINA (83)) - P X
CREATE FUNCTION CalculerNombreVoyages()
RETURNS INT
AS
BEGIN
RETURN (SELECT COUNT(*) FROM Voyages);
END;

```

25. Ecrire une procédure stockée qui permet d'afficher les pilotes qui ont piloté plus d'un pourcentage donné des avions de la compagnie (par exemple, plus de 20% des avions)

```

Query20.sql - ...:1080K\LOUBINA (70)) - P X SQLQuery19ex24.s...:1080K\LOUBINA (81)) - P X SQLQuery18ex23.s...:1080K\LOUBINA (83)) - P X
CREATE PROCEDURE PilotesPlusActifs (
@Pourcentage INT
)
AS
BEGIN
DECLARE @Seuil INT
SELECT @Seuil = (COUNT(*) * @Pourcentage) / 100 FROM Pilotes

SELECT P.Num_Pilote, P.Nom_Pilote, P.Prenom_Pilote
FROM Pilotes P
JOIN Vols V ON P.Num_Pilote = V.Code_Pilote
GROUP BY P.Num_Pilote, P.Nom_Pilote, P.Prenom_Pilote
HAVING COUNT(DISTINCT V.Code_Avion) > @Seuil
END;

```

26. Ecrire une procédure stockée qui permet d'ajouter les colonnes NbrAvions, NbrVoyages et Statut dans la table Pilotes et l'initialiser selon les cas suivants :

NbrVoyages : est le nombre des voyages réalisés par un pilote donné.

NbrAvions : Nombre des avions pilotés par un pilote donné.

Statut vaut :

- Expert : Si le pilote a piloté plus de 50% des avions de la compagnie.
- Qualifié : Si le pilote a piloté entre 50% et 5% des avions de la compagnie.
- Débitéur : Si le pilote a piloté moins de 5% des avions de la compagnie.

```
CREATE PROCEDURE InitializeDonnéesPilot
AS
BEGIN
    DECLARE @NbrAvions INT,
            @NbrVoyages INT;

    UPDATE Pilotes
    SET @NbrAvions = (SELECT COUNT(DISTINCT Code_Avion) FROM Vols WHERE Code_Pilote = Pilotes.Num_Pilote),
        @NbrVoyages = (SELECT COUNT(*) FROM vols WHERE Code_Pilote = Pilotes.Num_Pilote);

    UPDATE Pilotes SET NbrAvionss = @NbrAvions, NbrVoyagess = @NbrVoyages;

    UPDATE Pilotes
    SET Statut = CASE
        WHEN NbrAvions > (SELECT COUNT(*) * 0.5 FROM Avions) THEN 'Expert'
        WHEN NbrAvions BETWEEN (SELECT COUNT(*) * 0.05 FROM Avions) AND (SELECT COUNT(*) * 0.5 FROM Avions) THEN 'Qualifie'
        ELSE 'Débitéur'
    END;

END;
GO
EXEC InitializeDonnéesPilot;
```

27. Ecrire une procédure stockée qui permet de proposer tous les billets possibles (Classés par ordre décroissant des prix) pour une ville de départ et une ville d'arrivée données pour un nombre d'escale donné.

```
CREATE PROCEDURE billetsPossible
    @DepartVille VARCHAR(50),
    @villeArrivé VARCHAR(50),
    @NumEscale INT
AS
BEGIN
    SELECT B.Num_Billet, B.Num_Reservation, V.Num_Vol, V.Prix_Vol
    FROM Billets B
    INNER JOIN Ligne_Reservation LR ON B.Num_Reservation = LR.Num_Reservation
    INNER JOIN Vols V ON LR.Num_Vol = V.Num_Vol
    WHERE V.Ville_Départ = @DepartVille
    AND V.Ville_Arrivée = @villeArrivé
    AND (SELECT COUNT(*) - 1 FROM Ligne_Reservation WHERE Num_Reservation = B.Num_Reservation) = @NumEscale
    ORDER BY V.Prix_Vol DESC;
END;
GO
EXEC billetsPossible
    @DepartVille = 'Rabat',
    @villeArrivé = 'Tanger',
    @NumEscale = 0;
```

Num_Billet	Num_Reservation	Num_Vol	Prix_Vol
202	102	102	120

28. Ecrire un trigger qui permet de contrôler la disponibilité d'une place dans un avion pour un voyage et un passager donné (utiliser les deux fonctions, Complet qui teste si le voyage est complet et Occuper qui teste si le numéro de la place est occupée). Si le voyage n'est pas complet et la place est occupée le trigger propose le numéro d'une place disponible automatiquement.

```

CREATE TRIGGER disponibilitePlace
ON Voyages
AFTER INSERT
AS
BEGIN
    DECLARE @Num_Vol INT, @Num_Place INT;
    SELECT @Num_Vol = Num_Vol, @Num_Place = Num_Place FROM inserted;
    IF dbo.Complet(@Num_Vol) = 0
    BEGIN
        IF dbo.Occuper(@Num_Vol, @Num_Place) = 1
        BEGIN
            DECLARE @NewPlace INT;
            SELECT TOP 1 @NewPlace = Num_Place
            FROM Avions
            WHERE Num_Avion = (SELECT Code_Avion FROM Vols WHERE Num_Vol = @Num_Vol)
            UPDATE Voyages
            SET Num_Place = @NewPlace
            WHERE Num_Vol = @Num_Vol;
            PRINT 'La place occupée a été changée. Le nouveau numéro de place est ' ;
        END
    END
END;

```

29. Ecrire un trigger qui permet d'assurer l'insertion les noms et les prénoms des passagers en majuscule et de contrôler l'unicité de la clé.

```

CREATE TRIGGER UniqueEtUppercase ON Passagers
BEFORE INSERT
AS
BEGIN
    UPDATE Passagers
    SET Nom_Passager = UPPER(Nom_Passager),
        Pre_Passager = UPPER(Pre_Passager)
    FROM inserted
    WHERE Passagers.Code_Passager = inserted.Code_Passager;
    IF EXISTS (SELECT 1 FROM inserted ins
               JOIN Passagers p ON ins.Code_Passager = p.Code_Passager
               GROUP BY ins.Code_Passager
               HAVING COUNT(*) > 1)
    BEGIN
        PRINT 'Erreur : La clé n'est pas unique.';
        ROLLBACK TRANSACTION;
    END
END;

```

30. Ecrire un trigger qui permet de contrôler l'insertion d'un voyage pour un passager et son billet réservé avec le contrôle de pré-enregistrement du vol correspondant.

```

CREATE TRIGGER controlerInsertion
ON Voyages
BEFORE INSERT
AS
BEGIN
    DECLARE @Code_Passager INT, @Num_Billet INT, @Num_Vol INT;
    SELECT @Code_Passager = Code_Passager, @Num_Billet = Num_Billet, @Num_Vol = Num_Vol FROM inserted;
    IF NOT EXISTS (SELECT * FROM Billets WHERE Num_Billet = @Num_Billet AND Code_Passager = @Code_Passager)
    BEGIN
        PRINT 'Erreur : Le passager n'a pas de billet réservé pour ce voyage.';
    END
    ELSE
    BEGIN
        IF NOT EXISTS (SELECT * FROM Vols WHERE Num_Vol = @Num_Vol AND Pre_enregistrement = 1)
        BEGIN
            PRINT 'Erreur : Le vol correspondant n'est pas pré-enregistré.';
        END
        ELSE
        BEGIN
            INSERT INTO Voyages (Code_Passager, Num_Billet, Num_Vol, Num_Place)
            VALUES (@Code_Passager, @Num_Billet, @Num_Vol, (SELECT MAX(Num_Place) + 1 FROM Voyages WHERE Num_Vol = @Num_Vol));
            PRINT 'Insertion du voyage effectuée avec succès.';
        END
    END
END;

```

31. Ecrire un trigger qui permet la mise à jour le Statut et le NbrAvions et NbrVoyages correspondant a un pilote automatiquement dès l'insertion d'une ligne dans la table Voyages.

```

CREATE TRIGGER UpdatePilot
ON Voyages
AFTER INSERT
AS
BEGIN
    DECLARE @Code_Pilote INT;
    SELECT @Code_Pilote = Code_Pilote FROM inserted;
    UPDATE Pilotes
    SET NbrAvions = (SELECT COUNT(DISTINCT Code_Avion) FROM Vols WHERE Code_Pilote = @Code_Pilote),
        NbrVoyages = (SELECT COUNT(*) FROM Voyages WHERE Code_Pilote = @Code_Pilote)
    WHERE Num_Pilote = @Code_Pilote;
    UPDATE Pilotes
    SET Statut = CASE
        WHEN NbrAvions > (SELECT COUNT(*) * 0.5 FROM Avions) THEN 'Expert'
        WHEN NbrAvions BETWEEN (SELECT COUNT(*) * 0.05 FROM Avions) AND (SELECT COUNT(*) * 0.5 FROM Avions) THEN 'Qualifie'
        ELSE 'Débiteur'
    END
    WHERE Num_Pilote = @Code_Pilote;
END;

```

32. Ecrire un trigger qui provoque une erreur à l'insertion d'un tuple dans la table Voyage si le nombre de places accordées dépasse la capacité de l'avion.

```

CREATE TRIGGER MaxPlaces
ON Voyages
INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT V.Num_Vol
        FROM inserted I
        JOIN Vols V ON I.Num_Vol = V.Num_Vol
        JOIN Avions A ON V.Code_Avion = A.Num_Avion
        GROUP BY V.Num_Vol
        HAVING COUNT(I.Num_place) > MAX(A.Nbr_Place)
    )
    BEGIN
        PRINT 'Erreur : Le nombre de places accordées dépasse la capacité de l''avion.';
    END
END;

```

33. Ecrire un trigger qui mémorise dans une table quel utilisateur a fait, à quelle heure, une modification (insert, delete e ou update) et quelles nouvelles valeurs d'attributs il a insérées dans la table Reservations.

```

CREATE TRIGGER LogReservationChanges
ON Reservations
AFTER INSERT, DELETE, UPDATE
AS
BEGIN
    DECLARE @TypeAction VARCHAR(10);
    DECLARE @Usererr VARCHAR(100);
    DECLARE @Temp DATETIME;
    DECLARE @NewValues NVARCHAR(MAX);
    IF EXISTS (SELECT * FROM inserted)
        SET @TypeAction = 'INSERT';
    ELSE IF EXISTS (SELECT * FROM deleted)
        SET @TypeAction = 'DELETE';
    ELSE
        SET @TypeAction = 'UPDATE';
    SET @Usererr = SUSER_SNAME();
    SET @Temp = GETDATE();
    INSERT INTO ReservationChangeLog (TypeAction, Usererr, Temp, NewValues)
    VALUES (@TypeAction, @Usererr, @Temp, @NewValues);
END;

```

34. Créer un déclencheur qui permet la suppression en cascade d'un passager donnée. (Avec la mise à jour des tables en correspondances).

```

CREATE TRIGGER CascadeDeletePassenger
ON Passagers
AFTER DELETE
AS
BEGIN
    DELETE FROM Reservations WHERE Code_Passager IN (SELECT Code_Passager FROM deleted);
    DELETE FROM Ligne_Reservation WHERE Num_Reservation IN (SELECT Num_Reservation FROM deleted);
    DELETE FROM Billets WHERE Num_Reservation IN (SELECT Num_Reservation FROM deleted);
    DELETE FROM Voyages WHERE Code_Passager IN (SELECT Code_Passager FROM deleted);
END;

```

35. Ecrire un trigger qui permet la suppression en cascade d'un passager donnée. (Avec la mise à jour des tables en correspondances).

```

CREATE TRIGGER CascadeDeletePassenger
ON Passagers
AFTER DELETE
AS
BEGIN
    DELETE FROM Reservations WHERE Code_Passager IN (SELECT Code_Passager FROM deleted);
    DELETE FROM Ligne_Reservation WHERE Num_Reservation IN (SELECT Num_Reservation FROM deleted);
    DELETE FROM Billets WHERE Num_Reservation IN (SELECT Num_Reservation FROM deleted);
    DELETE FROM Voyages WHERE Code_Passager IN (SELECT Code_Passager FROM deleted);
END;

```

36. Ecrire un trigger qui consiste à corriger à la volée des saisies ou des modifications incorrectes. Tous les caractères de séparation de numéro de tel du passager que soit le tiret ou l'espace d'un numéro de téléphone devra être convertis en point et le caractère 'O' par celui de '0' (Zéro).

```

CREATE TRIGGER CorrectionNumero
ON Passagers
BEFORE INSERT, UPDATE
AS
BEGIN
    UPDATE Passagers
    SET Num_Tel = REPLACE(REPLACE(REPLACE(Num_Tel, '-', '.'), ' ', '.'), 'O', '0')
    WHERE Code_Passager IN (SELECT Code_Passager FROM inserted);
    IF EXISTS (SELECT * FROM deleted)
    BEGIN
        UPDATE Passagers
        SET Nom_Passager = i.Nom_Passager,
            Pre_Passager = i.Pre_Passager,
            Num_Passport = i.Num_Passport,
            Categorie = i.Categorie,
            Num_Tel = i.Num_Tel
        FROM inserted i
        WHERE Passagers.Code_Passager = i.Code_Passager;
    END
    ELSE
    BEGIN
        INSERT INTO Passagers (Code_Passager, Nom_Passager, Pre_Passager, Num_Passport, Categorie, Num_Tel)
        SELECT Code_Passager, Nom_Passager, Pre_Passager, Num_Passport, Categorie, Num_Tel
        FROM inserted;
    END;
END;

```

37. Ecrire un trigger qui permet de contrôler la validité de saisie de format de la date de départ et date d'arrivée. La date est valide sauf si elle ne possède que les chiffres, le caractère '/' et au maximum 10 caractères. Si la date contient l'un des caractères 'O' ou 'Q' le trigger les remplace par le chiffre 0.

```

CREATE TRIGGER CheckDateValidity
ON Vols
AFTER INSERT, UPDATE
AS
BEGIN
    UPDATE Vols
    SET Date_Depart = REPLACE(REPLACE(Date_Depart, 'O', '0'), 'Q', '0')
    WHERE LEN(Date_Depart) <= 10;
    UPDATE Vols
    SET Date_Arrivee = REPLACE(REPLACE(Date_Arrivee, 'O', '0'), 'Q', '0')
    WHERE LEN(Date_Arrivee) <= 10;
END;

```


38. Ecrire un trigger qui permet d'archiver toutes les opérations de suppression sur la table Voyages (Créer la table nécessaire pour l'archive).

```
CREATE TRIGGER ArchiveVoyages
ON Voyages
AFTER DELETE
AS
BEGIN
INSERT INTO VoyageArchive (Num_Vol, Date_Depart, Heure_Depart, Ville_Depart, Ville_Arrivee,
                           Code_Avion, Code_Pilote, Prix_Vol, Date_Suppression)
SELECT Num_Vol, Date_Depart, Heure_Depart, Ville_Depart, Ville_Arrivee, Code_Avion, Code_Pilote, Prix_Vol, GETDATE()
FROM deleted;
END;
```

39. Ecrire un trigger qui permet d'archiver la suppression des réservations selon la nature de traitement de la réservation soit annulée ou validée. Une réservation validée ne peut être supprimée qu'après 10 jours de la date de voyage et une réservation annuler ne peut être supprimer qu'après la date de validation (Créer la table nécessaire pour l'archive)

```
CREATE TRIGGER ArchiveDeletedReservations
ON Reservations
AFTER DELETE
AS
BEGIN
DECLARE @DeletedReservations TABLE (
    Num_Reservation INT,
    Date_Reservation DATE,
    Date_Validation DATE,
    Etat_Reservation VARCHAR(50),
    Code_Agence INT,
    Code_Passager INT,
    Prix_Total DECIMAL(10, 2)
);

INSERT INTO @DeletedReservations (Num_Reservation, Date_Reservation, Date_Validation, Etat_Reservation,
                                  Code_Agence, Code_Passager, Prix_Total)
SELECT Num_Reservation, Date_Reservation, Date_Validation, Etat_Reservation, Code_Agence, Code_Passager, Prix_Total
FROM deleted;
DECLARE @CurrentDate DATE;
SET @CurrentDate = GETDATE();
INSERT INTO ReservationArchive (Num_Reservation, Date_Reservation, Date_Validation, Etat_Reservation,
                                Code_Agence, Code_Passager, Prix_Total, Date_Suppression)
SELECT Num_Reservation, Date_Reservation, Date_Validation, Etat_Reservation, Code_Agence, Code_Passager, Prix_Total, @CurrentDate
FROM @DeletedReservations
WHERE
    (Etat_Reservation = 'Validée' AND DATEDIFF(DAY, Date_Reservation, @CurrentDate) >= 10)
    OR
    (Etat_Reservation = 'Annulée' AND Date_Suppression >= Date_Validation);
END;
```

40. Créer la vue ReservationValidees à partir de la table réservations pour avoir dans cette vue uniquement les réservations validées de l'agence 001. Les attributs de la vue seront : idzone, type, caract, dist. Cette vue est-elle modifiable ? Si vous pensez qu'elle n'est pas modifiable, comment la rendre modifiable

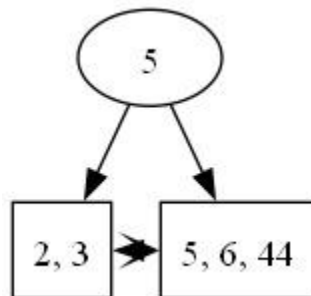
```

CREATE VIEW ReservationValidees AS
SELECT idzone, type, caract, dist
FROM Reservations
WHERE Etat_Reservation = 'Validée' AND Code_Agence = '001';

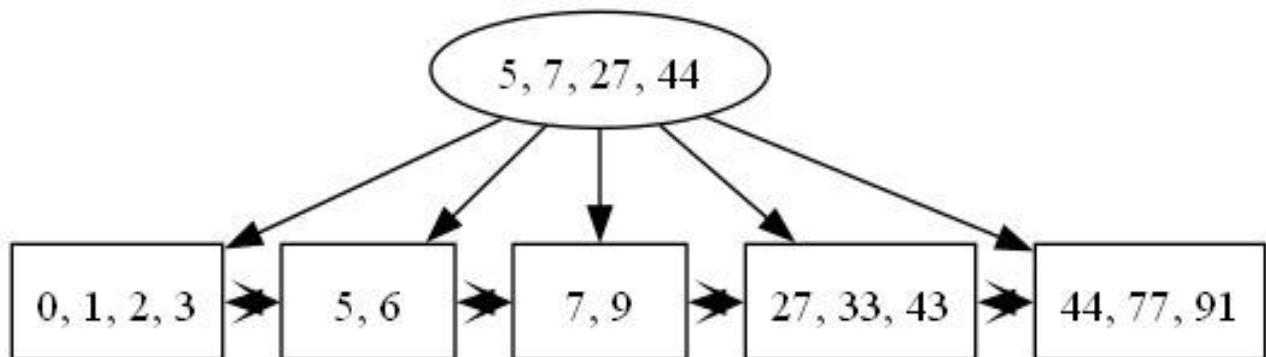
```

41. L'ajout dans un arbre B+ : Schématisé et décrire l'insertion étape par étape dans un arbre B+ de 3 clés la liste des valeurs suivantes : 3-6-44-5-2-77-1-7-9-91-33-43-0-27-88-55-54-56-57-52-44-24-25-26-27-98-99-4-6-8

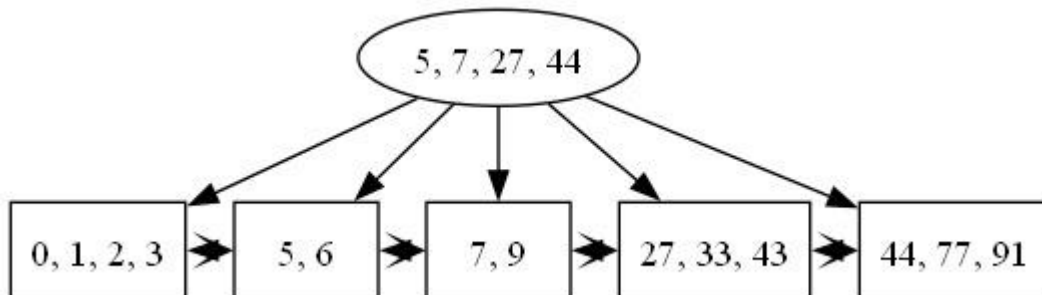
L'arbre B+ : Schématisé en utilisant la bibliothèque **Graphviz de python**



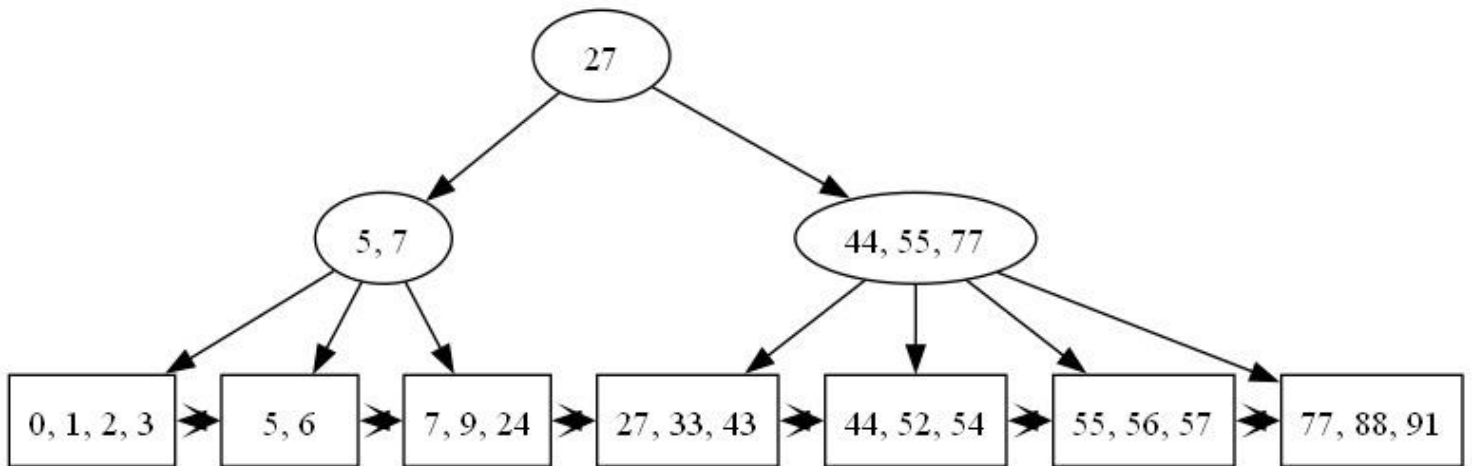
1- L'insertion de :[3, 6, 44, 5, 2] nous donne :



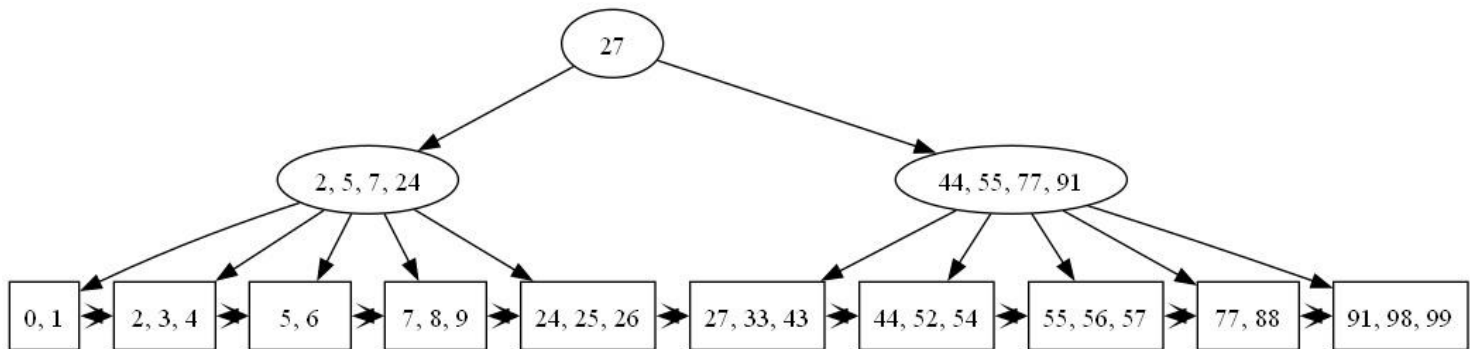
2- L'insertion de 77-1-7-9-91-33-43-0-27 :



3- L'insertion de 88-55-54-56-57-52-44-24 :

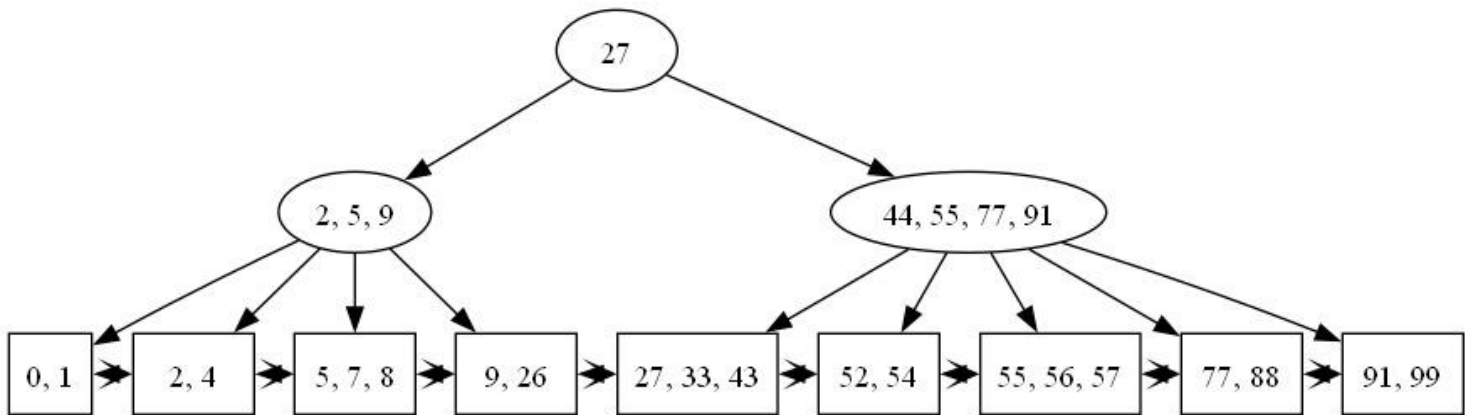


4- L'insertion de 25, 26, 27, 98, 99, 4, 6, 8 :

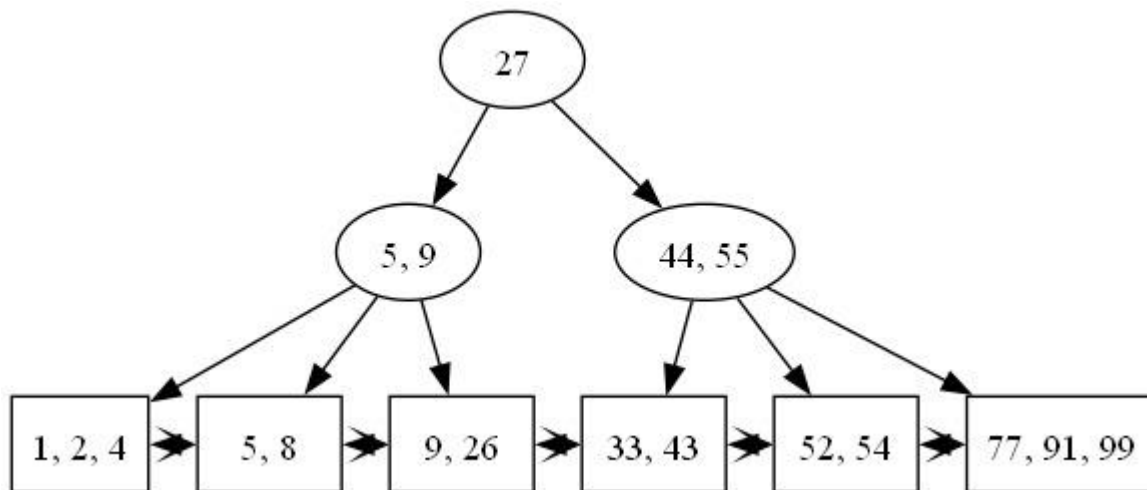


42. Suppression de l'arbre B+ : Schématisé et décrire la suppression étape par étape de l'arbre B+ (Question 44) des valeurs suivantes : 3-24-25-98-6-44-0-27-88-55-7-56-57-77-9-91-33-43-5-52-44-26-27-99-54-4-6-8-2-1.

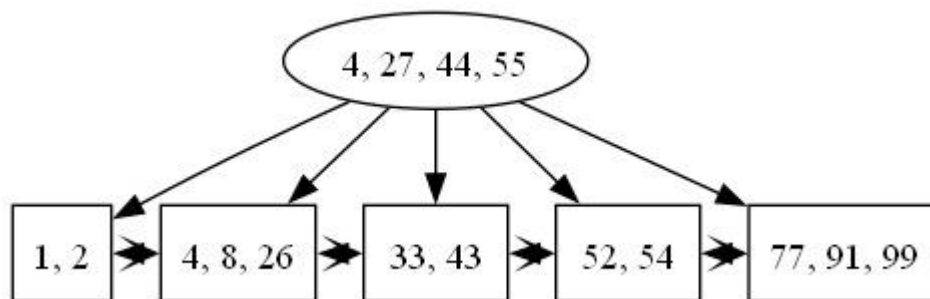
1- suppression de [3,24,25,98,6,44] :



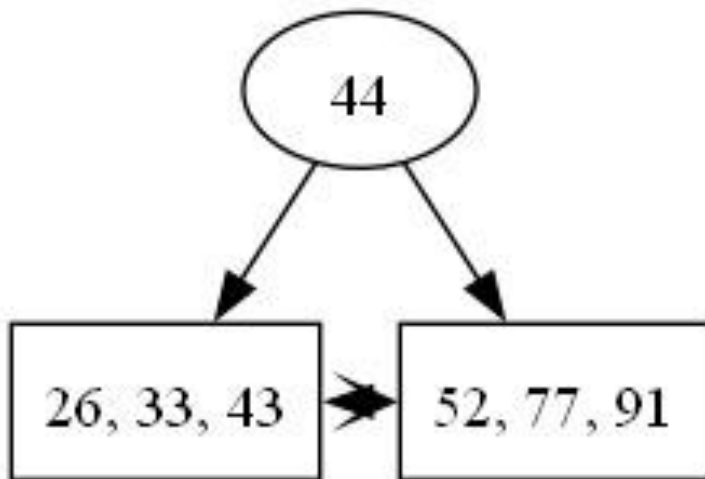
2- suppression de [0,88,55,7,56,57] :



3- suppression de [5,9] :



4- En fin ,suppression de 99-54-4-6-8-2-1 :



3. REALISATION DU PROJET BDD

3.1. But de ce projet

Le but de ce projet est de développer une application qui met en œuvre différents algorithmes d'indexation, notamment les algorithmes d'indexation par hachage et les arbres B+. L'objectif est de comprendre en profondeur le fonctionnement de ces algorithmes et leur impact sur les performances des bases de données relationnelles.

Plus précisément, les principaux objectifs de ce projet sont les suivants :

-Implémentation des Algorithmes d'Hachage : Nous avons mis en place des structures de données basées sur les algorithmes d'hachage, notamment le hachage linéaire et le hachage par division. Ces techniques seront utilisées pour organiser les données de manière efficace et pour accélérer les opérations de recherche et de récupération.

-Utilisation des Index B+ : Les avantages des arbres B+ en tant que structures d'indexation. Nous avons mis en œuvre ces structures pour améliorer les performances des requêtes en permettant un accès rapide et efficace aux données.

-Création d'une Interface de Démo : Nous avons développé une interface pratique qui permettra de visualiser les différences de performances entre les différentes méthodes d'indexation. Cette application fournira également des fonctionnalités pour créer, manipuler et interroger un exemple de base de données, ce qui permettra de mettre en évidence l'impact des index sur les temps de traitement des requêtes.

3.2. Outils utilisés

Pour réaliser cette interface , on a utilisé de différents outils tels que :

Python: Langage de programmation principal utilisé pour le développement de l'application.

Tkinter: Bibliothèque graphique standard de Python pour créer des interfaces utilisateur.

Microsoft SQL Server: Système de gestion de base de données relationnelle utilisé pour stocker et gérer les données de l'application.

Graphviz : un outil programmable qui affiche des graphes au format DOT et permet à l'utilisateur de faire des actions avec la souris .Utilisé dans notre interface pour afficher les arbres B+ .

Algorithmes d'hachage :

Chaînage séparé

Double hachage

Essai linéaire

Algorithmes d'indexation :

Indexe dense

Indexe creux

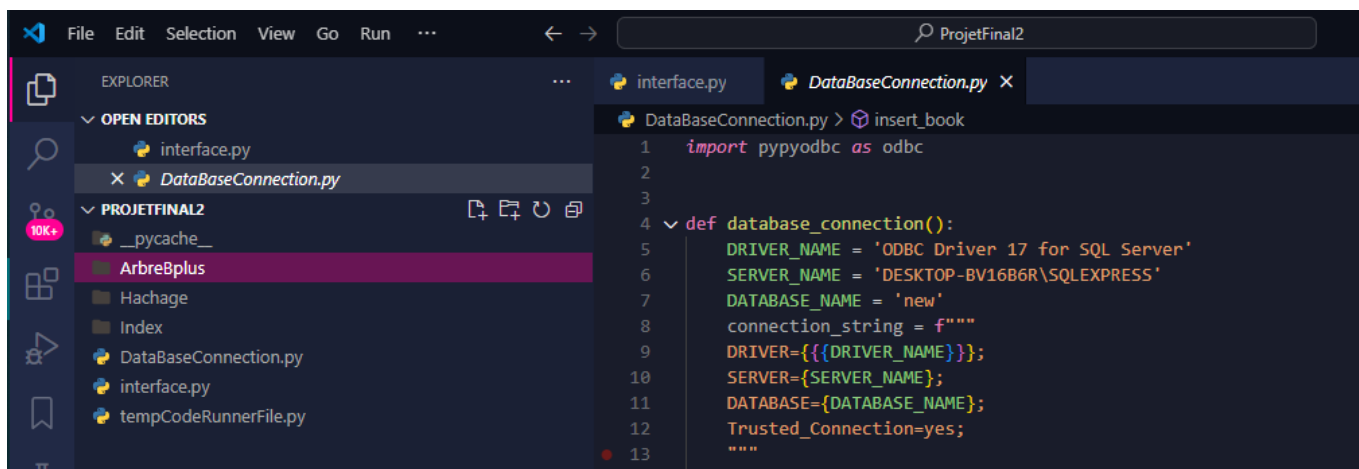
Arbres B+: Utilisés pour l'indexation et la recherche efficace des données.

3.3. Solution

On a réalisé une interface graphique en utilisant la bibliothèque de **Python Tkinter** pour implémenter les différents algorithmes d'hachage et indexation .

On a créé une base de données (**Bibliothèque de livres**) pour que l'utilisateur teste ces algorithmes d'une manière facile .

Notre projet est décomposé comme suit :



Le fichier interface

Page principale ,qui affiche tous les éléments de notre interface .

Pour l'exécuter , on doit taper la commande : **python interface.py**

Le fichier DatabaseConnection

Ce fichier élabore la connexion entre l'interface et notre base de données créé sous SQL SERVER .

Vous devez changer le nom du Server par le vôtre pour que la connexion s'effectue.

Les dossiers ArbresBplus, Hachage, Index

Ces dossiers contiennent les modules de tous les algorithmes déployés , ces modules sont appelés dans le fichier principale Interface.py .

3.4 Démonstration

Dans cette section , on va démontrer en images chaque partie de l'application réalisé :

La page Accueil :

Cette page affiche la base de données qu'on a créé pour tester les algorithmes de notre projet .

[Acceuil](#)[Indexation](#)[Hachage](#)[Arbres](#)

ACCEUIL

Bibliothèque des livres :

ID	Title	Author	Publication Year	Genre
1	The Great Gatsby	F. Scott Fitzgerald	1925	Classic
2	To Kill a Mockingbird	Harper Lee	1960	Fiction
3	1984	George Orwell	1949	Dystopian
4	Pride and Prejudice	Jane Austen	1813	Romance
5	The Catcher in the Rye	J.D. Salinger	1951	Coming-of-Age
6	The Lord of the Rings	J.R.R. Tolkien	1954	Fantasy
7	The Hitchhikers Guide to the Galaxy	Douglas Adams	1979	Science Fiction
8	The Da Vinci Code	Dan Brown	2003	Mystery
9	Harry Potter and the Philosophers Stone	J.K. Rowling	1997	Fantasy

La page Hachage :

Cette page affiche l'application de l'algorithme d'hachage.

L'utilisateur a le choix de choisir quel type à utiliser (Essai linéaire , Double hachage , Chainage séparé)

Chaque type saisi s'applique selon son algorithme dans le module (Hachage.py)

1. **Essai linéaire :**

- Lors de l'insertion d'un élément dans la table de hachage, si la position calculée par la fonction de hachage est déjà occupée, on recherche la prochaine position libre en avançant linéairement dans la table jusqu'à trouver une position vide.
- Cela signifie que si la position calculée est occupée, on continue à chercher la position suivante dans la table jusqu'à ce qu'on trouve une case vide.
- Cela peut conduire à des phénomènes de clustering (agglutination) où des collisions répétées regroupent plusieurs éléments dans des régions spécifiques de la table.

2. **Double hachage :**

- Cette technique utilise deux fonctions de hachage. Lorsqu'une collision se produit, elle utilise la seconde fonction de hachage pour calculer un décalage supplémentaire pour trouver une nouvelle position.
- Si la première fonction de hachage calcule la même position pour deux éléments, la deuxième fonction de hachage est utilisée pour calculer un autre emplacement, permettant de réduire les risques de clustering et d'améliorer la distribution des éléments dans la table de hachage.
- Le deuxième hachage doit être choisi avec soin pour garantir une bonne distribution des éléments.

3. **Chainage séparé :**

- Plutôt que de stocker directement les éléments dans la table de hachage, chaque emplacement de la table contient une liste (ou une autre structure de données comme un arbre) de tous les éléments qui se sont hachés à cet emplacement.
- Lorsqu'une collision se produit, l'élément nouvellement inséré est simplement ajouté à la liste des éléments déjà présents à cet emplacement.
- Cette méthode garantit que chaque emplacement de la table peut contenir plusieurs éléments sans nécessiter de recherche supplémentaire pour trouver un emplacement vacant, mais elle peut nécessiter plus de mémoire et peut être moins efficace en termes de temps d'accès si les listes deviennent très longues.

Exemples :

BDDPROJECT

Accueil **Indexation** **Hachage** **Arbres**

Algorithme: Hachage

Choisissez le type de l'algorithme souhaité:

Essai linéaire

Ent: Double hachage

Chaînage séparé

Entrer l'auteur :

Entrer l'année de publication :

Entrer le genre :

Ajouter

Rechercher par ID :

Rechercher Supprimer

h(x)	L'ELEMENT
------	-----------

Affichage de la page d'hachage avant le choix du type de traitement des données .

BDDPROJECT

Accueil Indexation Hachage Arbres

Algorithme: Hachage

Choisissez le type de l'algorithme souhaité:

Essai linéaire

Entrer le titre :

Entrer l'auteur :

Entrer l'année de publication :

Entrer le genre :

Ajouter

Rechercher par ID :

10

h(x)	L'ELEMENT
0	(10, 'The Alchemist', 'Paulo Coelho', '1988', 'Fiction')
1	(1, 'The Great Gatsby', 'F. Scott Fitzgerald', '1925', 'Classic')
2	(2, 'To Kill a Mockingbird', 'Harper Lee', '1960', 'Fiction')
3	(3, '1984', 'George Orwell', '1949', 'Dystopian')
4	(4, 'Pride and Prejudice', 'Jane Austen', '1813', 'Romance')
5	(6, 'The Lord of the Rings', 'J.R.R. Tolkien', '1954', 'Fantasy')
6	(7, 'The Hitchhikers Guide to the Galaxy', 'Douglas Adams', '1979', 'Science Fiction')
7	(8, 'The Da Vinci Code', 'Dan Brown', '2003', 'Mystery')
8	(9, 'Harry Potter and the Philosophers Stone', 'J.K. Rowling', '1997', 'Fantasy')

Ici on a un exemple de suppression en utilisant l'essai linéaire , Si on supprime par exemple la clé 10

BDDPROJECT

Accueil Indexation Hachage Arbres

Algorithme: Hachage

Choisissez le type de l'algorithme souhaité:

Essai linéaire

Entrer le titre :

Entrer l'auteur :

Entrer l'année de publication :

Entrer le genre :

Ajouter

Rechercher par ID :

h(x)	L'ELEMENT
0	(15, 'The Road', 'Cormac McCarthy', '2006', 'Post-Apocalyptic')
1	(1, 'The Great Gatsby', 'F. Scott Fitzgerald', '1925', 'Classic')
2	(2, 'To Kill a Mockingbird', 'Harper Lee', '1960', 'Fiction')
3	(3, '1984', 'George Orwell', '1949', 'Dystopian')
4	(4, 'Pride and Prejudice', 'Jane Austen', '1813', 'Romance')
5	(6, 'The Lord of the Rings', 'J.R.R. Tolkien', '1954', 'Fantasy')
6	(7, 'The Hitchhikers Guide to the Galaxy', 'Douglas Adams', '1979', 'Science Fiction')
7	(8, 'The Da Vinci Code', 'Dan Brown', '2003', 'Mystery')
8	(9, 'Harry Potter and the Philosophers Stone', 'J.K. Rowling', '1997', 'Fantasy')

Du même pour le chaînage séparé et le double hachage , toutes les fonctions d'ajout, suppression , recherche fonctionnent selon le type d'algorithme sélectionné .

BDDPROJECT

Accueil Indexation Hachage Arbres

Algorithme: Hachage

Choisissez le type de l'algorithme souhaité:

Chaînage séparé

Entrer le titre :

Entrer l'auteur :

Entrer l'année de publication :

Entrer le genre :

Ajouter

Rechercher par ID : Rechercher Supprimer

h(x)	L'ELEMENT
0	[(5, 'The Catcher in the Rye', 'J.D. Salinger', '1951', 'Coming-of-Age'), (10, 'The Alchemist', 'Paulo Coelho', '2000', 'Fiction')]
1	[(1, 'The Great Gatsby', 'F. Scott Fitzgerald', '1925', 'Classic'), (6, 'The Lord of the Rings', 'J.R.R. Tolkien', '1954', 'Fantasy')]
2	[(2, 'To Kill a Mockingbird', 'Harper Lee', '1960', 'Fiction'), (7, 'The Hitchhiker's Guide to the Galaxy', 'Douglas Adams', '1979', 'Science Fiction')]
3	[(3, '1984', 'George Orwell', '1949', 'Dystopian'), (8, 'The Da Vinci Code', 'Dan Brown', '2003', 'Mystery')]
4	[(4, 'Pride and Prejudice', 'Jane Austen', '1813', 'Romance'), (9, 'Harry Potter and the Philosopher's Stone', 'J.K. Rowling', '1997', 'Fantasy')]
5	[(5, 'The Catcher in the Rye', 'J.D. Salinger', '1951', 'Coming-of-Age')]
6	[(6, 'The Lord of the Rings', 'J.R.R. Tolkien', '1954', 'Fantasy')]
7	[(7, 'The Hitchhiker's Guide to the Galaxy', 'Douglas Adams', '1979', 'Science Fiction')]
8	[(8, 'The Da Vinci Code', 'Dan Brown', '2003', 'Mystery')]

BDDPROJECT

Accueil Indexation Hachage Arbres

Algorithme: Hachage

Choisissez le type de l'algorithme souhaité:

Double hachage

Entrer le titre :

Entrer l'auteur :

Entrer l'année de publication :

Entrer le genre :

Ajouter

Rechercher par ID : Rechercher Supprimer

h(x)	L'ELEMENT
0	(5, 'The Catcher in the Rye', 'J.D. Salinger', '1951', 'Coming-of-Age')
1	(1, 'The Great Gatsby', 'F. Scott Fitzgerald', '1925', 'Classic')
2	(2, 'To Kill a Mockingbird', 'Harper Lee', '1960', 'Fiction')
3	(3, '1984', 'George Orwell', '1949', 'Dystopian')
4	(4, 'Pride and Prejudice', 'Jane Austen', '1813', 'Romance')
5	(6, 'The Lord of the Rings', 'J.R.R. Tolkien', '1954', 'Fantasy')
6	(7, 'The Hitchhiker's Guide to the Galaxy', 'Douglas Adams', '1979', 'Science Fiction')
7	(8, 'The Da Vinci Code', 'Dan Brown', '2003', 'Mystery')
8	(9, 'Harry Potter and the Philosopher's Stone', 'J.K. Rowling', '1997', 'Fantasy')

La page Indexation :

L'utilisateur a le choix d'utiliser l'un des deux algorithmes définis :

- **Indexation dense :**

- Chaque élément de la structure d'index correspond à un élément du tableau.
- Cela signifie qu'il y a une entrée dans l'index pour chaque élément qu'il soit utilisé ou non.
- Les index denses sont utilisés lorsque l'espace mémoire n'est pas une préoccupation majeure et que l'accès aux éléments doit être rapide et prévisible.

- **Indexation creuse :**

- À l'inverse, dans une indexation creuse, seuls les éléments actuellement présents ou significatifs dans la table sont inclus dans l'index.
- Les emplacements correspondant aux éléments absents sont souvent laissés vides ou marqués comme tels, ce qui économise de l'espace mémoire.

Affichage de la page d'indexation avant le choix d'utilisateur

BDDPROJECT

Accueil **Indexation** **Hachage** **Arbres**

Algorithme : Indexation

Choisissez le type de l'algorithme souhaité:

Indexe Dense
Indexe Dense
Indexe Creux

Entrer le titre :

Entrer l'auteur :

Entrer l'année de publication :

Entrer le genre :

Ajouter

Rechercher par Année: **Rechercher** **Supprimer**

Année de publication	L'ELEMENT
----------------------	-----------

Exemple d'insertion (Index dense)

BDDPROJECT

AcceuilIndexationHachageArbres

Algorithme : Indexation

Choisissez le type de l'algorithme souhaité:

Indexe Dense

Entrer le titre : AISD

Entrer l'auteur : AISD

Entrer l'année de publication : 2023

Entrer le genre : AISD

Ajouter

Rechercher par Année:

Rechercher Supprimer

Année de publication	L'ELEMENT
2006	15 {The Road} {Cormac McCarthy} 2006 Post-Apocalyptic
2003	16 {The Kite Runner} {Khaled Hosseini} 2003 {Historical Fiction}
2005	17 {The Girl with the Dragon Tattoo} {Stieg Larsson} 2005 Crime
1985	18 {The Handmaids Tale} {Margaret Atwood} 1985 Dystopian
1937	19 {The Hobbit} {J.R.R. Tolkien} 1937 Fantasy
2000	20 IIII Im 2000 klm
2000	21 new new 2000 new
2005	22 hanan hanan 2005 hanan
1900	23 nn nn 1900 nn

Résultat :

BDDPROJECT

Acceuil

Indexation

Hachage

Arbres

Algorithme : Idexation

Choisissez le type de l'algorithme souhaité:

Indexe Dense

Entrer le titre :

Entrer l'auteur :

Entrer l'année de publication :

Entrer le genre :

Ajouter

Rechercher par Année:

Rechercher

Supprimer

Année de publication	L'ELEMENT
2003	16 {The Kite Runner} {Khaled Hosseini} 2003 {Historical Fiction}
2005	17 {The Girl with the Dragon Tattoo} {Stieg Larsson} 2005 Crime
1985	18 {The Handmaids Tale} {Margaret Atwood} 1985 Dystopian
1937	19 {The Hobbit} {J.R.R. Tolkien} 1937 Fantasy
2000	20 IIII Im 2000 klm
2000	21 new new 2000 new
2005	22 hanan hanan 2005 hanan
1900	23 nn nn 1900 nn
2023	24 AISD AISD 2023 AISD

Dans notre base de données :

100 %

Results Messages

	id	Title	Author	Year	Genre
14	14	The Hunger Games	Suzanne Collins	2008	Science Fiction
15	15	The Road	Comac McCarthy	2006	Post-Apocaly...
16	16	The Kite Runner	Khaled Hosseini	2003	Historical Ficti...
17	17	The Girl with the Dragon Tattoo	Stieg Larsson	2005	Crime
18	18	The Handmaids Tale	Margaret Atwood	1985	Dystopian
19	19	The Hobbit	J.R.R. Tolkien	1937	Fantasy
20	20	III	Im	2000	klm
21	21	new	new	2000	new
22	22	hanan	hanan	2005	hanan
23	23	nn	nn	1900	nn
24	24	AI SD	AI SD	2023	AI SD

Exemple de Recherche : Année 2023

BDDPROJECT

Accueil Indexation Hachage Arbres

Algorithme : Indexation

Choisissez le type de l'algorithme souhaité:

Indexe Dense

Entrer le titre :

Entrer l'auteur :

Entrer l'année de publication :

Entrer le genre :

Ajouter

Rechercher par Année:

Rechercher Supprimer

Année de publication	L'ELEMENT
2023	24 AI SD AI SD 2023 AI SD

Exemple de suppression (2023)

BDDPROJECT

— □ ×

Accueil Indexation Hachage Arbres

Algorithme : Indexation

Choisissez le type de l'algorithme souhaité:

Indexe Dense ▾

Entrer le titre :

Entrer l'auteur :

Entrer l'année de publication :

Entrer le genre :

Ajouter

Rechercher par Année:

2023

Rechercher Supprimer

Année de publication	L'ELEMENT
2006	15 {The Road} {Cormac McCarthy} 2006 Post-Apocalyptic
2003	16 {The Kite Runner} {Khaled Hosseini} 2003 {Historical Fiction}
2005	17 {The Girl with the Dragon Tattoo} {Stieg Larsson} 2005 Crime
1985	18 {The Handmaids Tale} {Margaret Atwood} 1985 Dystopian
1937	19 {The Hobbit} {J.R.R. Tolkien} 1937 Fantasy
2000	20 IIII Im 2000 klm
2000	21 new new 2000 new
2005	22 hanan hanan 2005 hanan
1900	23 nn nn 1900 nn

La page Arbres :

Cette page affiche la page d 'arbre B+ :

Lorsque l'utilisateur insère ou supprimer une valeur (par année) , l'arbre s'affiche et se met à jour en même temps comme dans ces exemples :

BDDPROJECT

Acceuil

Indexation

Hachage

Arbres

Algorithme : Les arbres

Arbre B+ ▼

Entrer le titre :

Entrer l'auteur :


Entrer l'année de publication :

Entrer le genre :

Ajouter

Supprimer par Year : Supprimer

Cas de suppression :

 BDDPROJECT


Acceuil

Indexation

Hachage

Arbres

Algorithme : Les arbres

Arbre B+ 

Entrer le titre :

Entrer l'auteur :

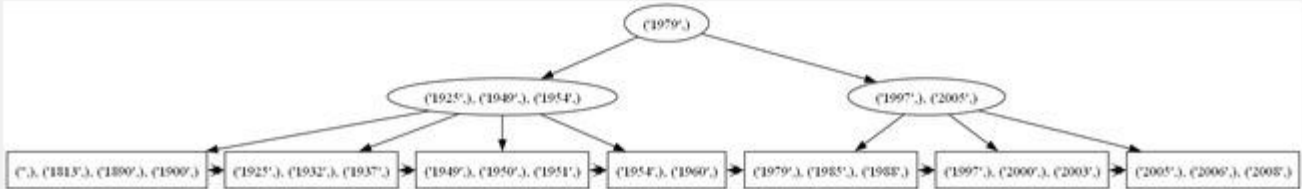
Entrer l'année de publication :

Entrer le genre :

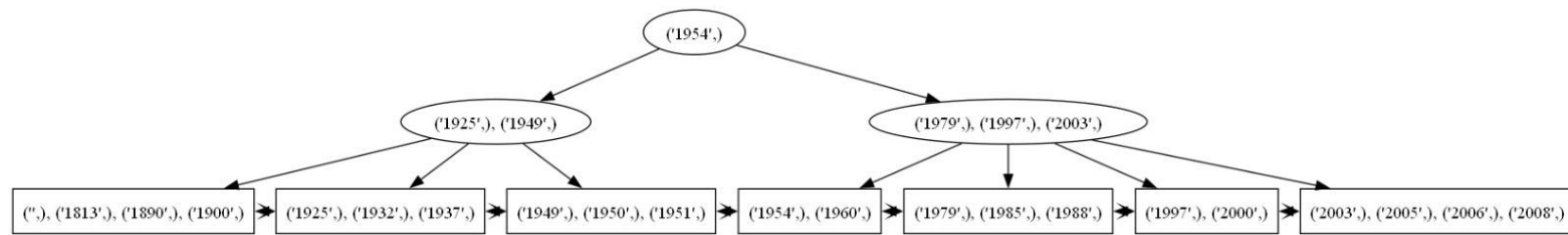
Ajouter

Supprimer par Year :

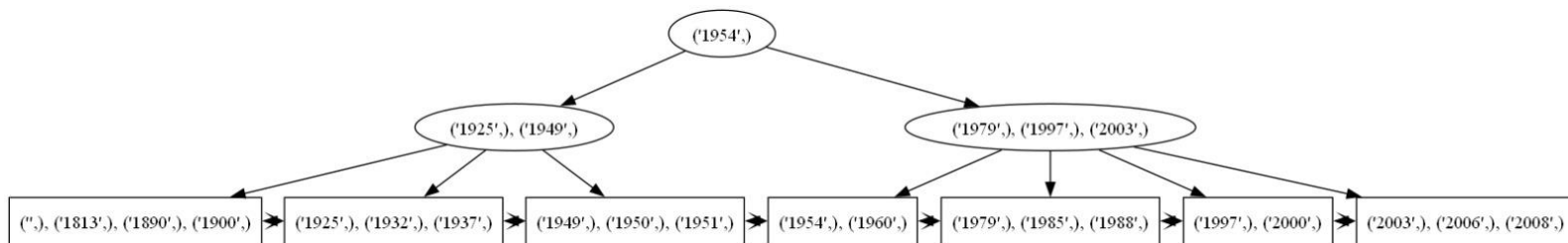
Supprimer



L'arbre généré (en ajoutant 2005) :



L'arbre généré après suppression(2005) :



4. CONCLUSION

Ce projet et ce TP ont été des expériences enrichissantes qui nous ont permis d'explorer en profondeur le domaine passionnant de la gestion de bases de données en utilisant Python et SQL Server.

Nous avons pu mettre en pratique nos connaissances théoriques en concevant et en mettant en œuvre une application qui offre une interface utilisateur conviviale pour interagir avec les données stockées dans une base de données SQL Server. L'utilisation de structures de données avancées telles que les arbres B+ pour l'indexation des données a été un élément crucial pour garantir des performances optimales lors des opérations de recherche et de manipulation des données.

Tout au long du projet et du TP, nous avons été confrontés à divers défis, notamment la gestion des connexions à la base de données, la création d'une interface utilisateur intuitive et la coordination entre les différentes couches de l'application. Cependant, grâce à une planification minutieuse, à une collaboration efficace et à une résolution de problèmes méthodique, nous avons pu surmonter ces obstacles avec succès.

En fin de compte, ce projet et ce TP ont abouti à la création d'une interface fonctionnelle et performante qui répond aux besoins de gestion de données de manière efficace et élégante. Ils représentent des étapes importantes dans notre parcours de développement et science de données, nous permettant d'approfondir nos connaissances en Python, en bases de données relationnelles et en développement d'interfaces utilisateur, tout en acquérant une expérience pratique précieuse avec l'un des outils les plus utilisés dans l'industrie, SQL Server.

5. WEBOGRAPHIE

<https://chat.openai.com/>

<https://docs.python.org/3/library/tk.html>

<https://forefront.ai>

<https://stackoverflow.com/>

<https://www.w3schools.com/sql/>

<https://docs.python.org/3/library/tk.html>

<https://graphviz.org/>