---------Rapport des TPS Ounissa ----------

----------TP 1----------

--- Pour le tp1, il fallait créer les tables demandées, après avoir créé la bade de données, que j'ai appelé BDD

---------J’ai inclus les contraintes de clés étrangères dès cet exercice, je ne savais pas que la question était prévue ailleurs

--- script de créarion des tables

```
CREATE TABLE Segment (
   indIP VARCHAR(11) NOT NULL PRIMARY KEY,
   nomSegment VARCHAR(20) NOT NULL,
   etage TINYINT
);

CREATE TABLE Salle (
   nSalle VARCHAR(7) NOT NULL PRIMARY KEY,
   nomSalle VARCHAR(20) NOT NULL,
   nbPoste TINYINT,
   indIP VARCHAR(11),
   FOREIGN KEY (indIP) REFERENCES Segment(indIP)
);

CREATE TABLE Poste (
   nPoste VARCHAR(7) NOT NULL PRIMARY KEY,
   nomPoste VARCHAR(20) NOT NULL,
   indIP VARCHAR(11),
   ad VARCHAR(3),
```

```
  CHECK (
    ad REGEXP '^[0-9]{1,3}$' AND CAST(ad AS UNSIGNED) BETWEEN 0 AND 255
  ),
  typePoste VARCHAR(9),
  nSalle VARCHAR(7),
  FOREIGN KEY (indIP) REFERENCES Segment(indIP),
  FOREIGN KEY (nSalle) REFERENCES Salle(nSalle)
);

CREATE TABLE Logiciel (
  nLog VARCHAR(5) NOT NULL PRIMARY KEY,
  nomLog VARCHAR(20) NOT NULL,
  dateAch DATETIME,
  version VARCHAR(7),
  typeLog VARCHAR(9),
  prix DECIMAL(6,2),
  CHECK (prix > 0)
);

CREATE TABLE Installer (
  numIns INT NOT NULL PRIMARY KEY,
  nPoste VARCHAR(7),
  nLog VARCHAR(5),
  dateIns TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  delai SMALLINT,
  FOREIGN KEY (nPoste) REFERENCES Poste(nPoste),
  FOREIGN KEY (nLog) REFERENCES Logiciel(nLog)
);
```

```
CREATE TABLE Types (
   typeLp VARCHAR(9) NOT NULL PRIMARY KEY,
   nomType VARCHAR(20)
);

--- afficher la descriptopn de toutes les tables

DESC Segment;
DESC Salle;
DESC Poste;
DESC Logiciel;
DESC Installer;
DESC Types;

--- script pour supprmier
--On doit respecter l'ordre de dépendance sinon on a des erreurs a la suppression

DROP TABLE IF EXISTS Installer, Poste, Salle, Logiciel, Segment, Types;

---------------------------------------
---------------------------------------
-------TP 2 -------------------------
---------------------------------------

----Script de remplissage

nsert into table Segment (indIP, nomSegment, etage) values
```

('130.120.80', 'Brin', 0),

('130.120.81', 'Brin', 1),

('130.120.82', 'Brin', 2);

insert into table Salle (nSalle, nomSalle, nbPoste, indIP) values

('s01', 'Salle 1', 3, '130.120.80'),

('s02', 'Salle 2', 2, '130.120.80'),

('s03', 'Salle 3', 2, '130.120.80'),

('s11', 'Salle 11', 2, '130.120.81'),

('s12', 'Salle 12', 1, '130.120.81'),

('s21', 'Salle 21', 2, '130.120.82'),

('s22', 'Salle 22', 0, '130.120.83'),

('s23', 'Salle 23', 0, '130.120.83');

insert into table Poste (nPoste, nomPoste, indIP, ad, typePoste, nSalle) values

('p1', 'Poste 1', '130.120.80', '01', 'TX', 's01'),

('p2', 'Poste 2', '130.120.80', '02', 'UNIX', 's01'),

('p3', 'Poste 3', '130.120.80', '03', 'TX', 's01'),

('p4', 'Poste 4', '130.120.80', '04', 'PCWS', 's02'),

('p5', 'Poste 5', '130.120.80', '05', 'PCWS', 's02'),

('p6', 'Poste 6', '130.120.80', '06', 'UNIX', 's03'),

('p7', 'Poste 7', '130.120.80', '07', 'TX', 's03'),

('p8', 'Poste 8', '130.120.81', '01', 'UNIX', 's11'),

('p9', 'Poste 9', '130.120.81', '02', 'TX', 's11'),

('p10', 'Poste 10', '130.120.81', '03', 'UNIX', 's12'),

('p11', 'Poste 11', '130.120.82', '01', 'PCNT', 's21'),

('p12', 'Poste 12', '130.120.82', '02', 'PCWS', 's21');

```
INSERT INTO TABLE Logiciel (nLog, nomLog, dateAch, version, typeLog, prix) values
('log1', 'oracle 6', '1995-05-13', 6.2, 'UNIX', 3000),
('log2', 'oracle 8', '1999-05-13', 6.2, 'UNIX', 3000),
('log3', 'SQL Server', '1995-05-13', 6.2, 'UNIX', 3000),
('log4', 'Front Page', '1995-05-13', 6.2, 'UNIX', 3000),
('log5', 'Windev', '1995-05-13', 6.2, 'UNIX', 3000),
('log6', 'SQL*NET', '1995-05-13', 6.2, 'UNIX', 3000),
('log7', 'IIS', '1995-05-13', 6.2, 'UNIX', 3000),
('log8', 'Dream Weaver', '1995-05-13', 6.2, 'UNIX', 3000);

insert into table Types (typeLp, nomType) values
('TX', 'Terminal X-windows'),
('UNIX', 'Système Unix'),
('PCNT', 'PC Windows NT'),
('PCWS', 'PC Windows'),
('NC', 'Network Computer');

--- 2. 2 sequence pour numIns, modification du script de création

---Script pour modifier les tables

ALTER TABLE Installer
MODIFY numIns INT NOT NULL AUTO_INCREMENT PRIMARY KEY;

INSERT INTO TABLE Installer (nPoste, nLog, dateIns) values
('p2', 'log1', '2003-05-15'),
('p2', 'log2', '2003-09-17'),
('p4', 'log5', NULL),
```

```
('p6', 'log6', '2003-05-20'),
('p6', 'log1', '2003-05-20'),
('p8', 'log2', '2003-05-19'),
('p8', 'log6', '2003-05-20'),
('p11', 'log3', '2003-04-20'),
('p12', 'log4', '2003-04-20'),
('p11', 'log7', '2003-04-20'),
('p7', 'log7', '2003-04-01');

ALTER TABLE Segment
ADD nbSalle TINYINT DEFAULT 0;

--mise à jour
UPDATE Segment s
SET nbSalle = (
   SELECT COUNT(*)
   FROM Salle sa
   WHERE sa.indIP = s.indIP
);

--- modification de la table salle
ALTER TABLE Salle
MODIFY nomSalle VARCHAR(30) NOT NULL;

ALTER TABLE Segment
MODIFY nomSegment VARCHAR(14) NOT NULL;

---contraite d'installation d'un unique logicile
```

```
ALTER TABLE Installer
ADD CONSTRAINT unique_installation
UNIQUE (nPoste, nLog);

----mise à jour de la suppression

SET FOREIGN_KEY_CHECKS = 0;

DROP TABLE IF EXISTS Installer, Poste, Salle, Logiciel, Segment, Types;

SET FOREIGN_KEY_CHECKS = 1;



---------------------------------------
---------------------------------------
-------TP 3  -------------------------
---------------------------------------

--- script createDynamqi

-- Création de la table Softs à partir de Logiciel
CREATE TABLE Softs AS
SELECT
   nomLog AS nomSoft,
   version,
   prix
FROM Logiciel;
```

```
-- Création de la table PCSeuls à partir de Poste (filtrée sur type 'PCWS' ou 'PCNT')
CREATE TABLE PCSeuls AS
SELECT
   nPoste AS nP,
   nomPoste AS nomP,
   indIP AS seg,
   ad,
   typePoste AS typeP,
   nSalle AS salle
FROM Poste
WHERE typePoste IN ('PCWS', 'PCNT');

--- pour exécuter le script

mysql -u ounissa -p BDD < creaDynamique.sql

----------------------
-------tp3-------------
----------------------

-- 1. Extraire le type du poste 'p8'
SELECT typePoste
FROM Poste
WHERE nPoste = 'p8';

-- 2. Extraire le nom des logiciels de type 'UNIX'
SELECT nomLog
```

```
FROM Logiciel
WHERE typeLog = 'UNIX';

-- 3. Noms, adresses IP, numéros de salle des postes de type 'UNIX' ou 'PCWS'
SELECT nomPoste, indIP AS adresseIP, nSalle
FROM Poste
WHERE typePoste IN ('UNIX', 'PCWS');

-- 4. Noms, adresses IP, numéros de salle des postes du segment '130.120.80', triés par numéro de salle décroissant
SELECT nomPoste, indIP AS adresseIP, nSalle
FROM Poste
WHERE indIP = '130.120.80'
ORDER BY nSalle DESC;

-- 5. Numéros des logiciels installés sur le poste 'p6'
SELECT nLog
FROM Installer
WHERE nPoste = 'p6';

-- 6. Numéros des postes qui hébergent le logiciel 'log1'
SELECT nPoste
FROM Installer
WHERE nLog = 'log1';

-- 7. Nom et adresse IP complète (concaténée) des postes de type 'TX'
SELECT
  nomPoste,
```

```
   CONCAT(indIP, '.', ad) AS adresseIPComplete
FROM Poste
WHERE typePoste = 'TX';




----nbr de logiciles installés
SELECT
   p.nPoste,
   COUNT(i.nLog) AS nbr_logiciels_installes
FROM Poste p
LEFT JOIN Installer i ON p.nPoste = i.nPoste
GROUP BY p.nPoste;

-- nbr de postes par salle
SELECT
   s.nSalle,
   COUNT(p.nPoste) AS nbr_postes
FROM Salle s
LEFT JOIN Poste p ON s.nSalle = p.nSalle
GROUP BY s.nSalle;

--- nbr installatiosn différentes
SELECT
   l.nLog,
   COUNT(DISTINCT i.nPoste) AS nbr_postes_diff
FROM Logiciel l
```

```
LEFT JOIN Installer i ON l.nLog = i.nLog
GROUP BY l.nLog;

-- moyenne des prix
SELECT AVG(prix) AS moyenne_prix_unix
FROM Logiciel
WHERE typeLog = 'UNIX';

-- date d'achat la plus recente
SELECT MAX(dateAch) AS date_achat_plus_recente
FROM Logiciel;

---numéro de poste avec  logiciels
SELECT
   i.nPoste
FROM Installer i
GROUP BY i.nPoste
HAVING COUNT(DISTINCT i.nLog) = 2;

--- numéro de postes avec 2 logicels num 2
SELECT COUNT(*) AS nbr_postes_2_logiciels
FROM (
   SELECT i.nPoste
   FROM Installer i
   GROUP BY i.nPoste
   HAVING COUNT(DISTINCT i.nLog) = 2
) AS postes_2_logiciels;
```

```
--types de postes non recincés
SELECT t.typeLp
FROM Types t
LEFT JOIN Poste p ON t.typeLp = p.typePoste
WHERE p.typePoste IS NULL;

--- à la fois type de poste et logiciel

SELECT DISTINCT t.typeLp
FROM Types t
JOIN Poste p ON t.typeLp = p.typePoste
JOIN Logiciel l ON t.typeLp = l.typeLog;

-- type de poste pas type de logiciel
SELECT DISTINCT p.typePoste
FROM Poste p
LEFT JOIN Logiciel l ON p.typePoste = l.typeLog
WHERE l.typeLog IS NULL;

--adresse ip des postes log6
SELECT CONCAT(p.indIP, '.', p.ad) AS adresse_ip_complete
FROM Poste p
JOIN Installer i ON p.nPoste = i.nPoste
WHERE i.nLog = 'log6';

--adresse ip des ppostes oracle 8
```

```
SELECT DISTINCT CONCAT(p.indIP, '.', p.ad) AS adresse_ip_complete
FROM Logiciel l
JOIN Installer i ON l.nLog = i.nLog
JOIN Poste p ON i.nPoste = p.nPoste
WHERE l.nomLog = 'oracle 8';

-- segments avec trois postes de travail 'tx'
SELECT s.nomSegment
FROM Segment s
JOIN Poste p ON s.indIP = p.indIP
WHERE p.typePoste = 'TX'
GROUP BY s.indIP, s.nomSegment
HAVING COUNT(*) = 3;

--salle avec un poste oracle 6
SELECT DISTINCT sa.nomSalle
FROM Salle sa
JOIN Poste p ON sa.nSalle = p.nSalle
JOIN Installer i ON p.nPoste = i.nPoste
JOIN Logiciel l ON i.nLog = l.nLog
WHERE l.nomLog = 'oracle 6';

--nom du logiciel installé le plus récent
SELECT nomLog
FROM Logiciel
WHERE dateAch = (SELECT MAX(dateAch) FROM Logiciel);
```

------TP MERISE----------

--- MCD 1

-- je n'arrivee pas à inclure les images dans le fichier, mais voici une représentation avec les carinalités

--- Client 1,N ——— posséder ——— 0,N Matériel

--- MPD 1

```
CREATE TABLE Produit (
   idProduit INT PRIMARY KEY AUTO_INCREMENT,
   nomProduit VARCHAR(50),
   typeProduit ENUM('animal', 'fruit', 'legume')
);

CREATE TABLE Vente (
   idVente INT PRIMARY KEY AUTO_INCREMENT,
   dateVente DATE
);

CREATE TABLE Vendre (
   idProduit INT,
   idVente INT,
   poidsKg DECIMAL(5,2),
   prixUnitaire DECIMAL(6,2),
```

```
  PRIMARY KEY (idProduit, idVente),
  FOREIGN KEY (idProduit) REFERENCES Produit(idProduit),
  FOREIGN KEY (idVente) REFERENCES Vente(idVente)
);

---EXEMPLE DE ECONTENU

-- Produits
INSERT INTO Produit (nomProduit, typeProduit) VALUES
('Lapin', 'animal'), ('Pomme', 'fruit'), ('Carotte', 'legume');

-- Vente
INSERT INTO Vente (dateVente) VALUES ('2025-05-30');

-- ProduitVendu
INSERT INTO ProduitVendu (idVente, idProduit, poidsKg, prixUnitaire) VALUES
(1, 1, 2.5, 7.50),  -- 2.5kg de Lapin
(1, 2, 1.0, 3.00);  -- 1kg de Pomme


--- MCD 2

-- je n'arrivee pas à inclure les images dans le fichier, mais voici une représentation avec les carinalités

/*
Client
```

→ (idClient, nom, adresse, téléphone)

Matériel

→ (idMateriel, marque, modele)

Intervention

→ (idIntervention, dateIntervention, dureeHeures, typeIntervention, prixHoraire)

Vente

→ (idVente, description, prix)

*/

--- Client (1,N) ——— possède ——— (0,1) Matériel

--- Matériel (1,1) ——— fait l'objet de ——— (0,N) Intervention

---Intervention (0,N) ——— inclut ——— (0,N) Vente

----MPD 2

```
-- Table Client
CREATE TABLE Client (
   idClient INT PRIMARY KEY,
   nom VARCHAR(50),
   adresse VARCHAR(100),
   telephone VARCHAR(15)
);

-- Table Materiel
```

```
CREATE TABLE Materiel (
   idMateriel INT PRIMARY KEY,
   marque VARCHAR(50),
   modele VARCHAR(50),
   idClient INT,
   FOREIGN KEY (idClient) REFERENCES Client(idClient)
);

-- Table Intervention
CREATE TABLE Intervention (
   idIntervention INT PRIMARY KEY,
   dateIntervention DATE,
   dureeHeures DECIMAL(4,2),
   typeIntervention VARCHAR(50),
   prixHoraire DECIMAL(6,2),
   idMateriel INT,
   FOREIGN KEY (idMateriel) REFERENCES Materiel(idMateriel)
);

-- Table Vente
CREATE TABLE Vente (
   idVente INT PRIMARY KEY,
   description VARCHAR(100),
   prix DECIMAL(7,2)
);

-- Table associative Intervention_Vente (pour l'association n-n "Inclure")
CREATE TABLE Intervention_Vente (
```

```
  idIntervention INT,
  idVente INT,
  PRIMARY KEY (idIntervention, idVente),
  FOREIGN KEY (idIntervention) REFERENCES Intervention(idIntervention),
  FOREIGN KEY (idVente) REFERENCES Vente(idVente)
);

-- EXEMPLE DE CONTENU
-- Clients
INSERT INTO Client (nom, adresse, telephone) VALUES
('Jean Dupont', '123 rue des Lilas', '0601020304');

-- Matériel
INSERT INTO Materiel (marque, modele, idClient) VALUES
('HP', 'Pavilion 15', 1);

-- Intervention
INSERT INTO Intervention (dateInterv, dureeHeures, typeIntervention, prixHoraire, idMateriel)
VALUES ('2025-05-29', 2.0, 'réinstallation système', 40.00, 1);

-- Vente de matériel dans une intervention
INSERT INTO VenteMateriel (idIntervention, description, prix)
VALUES (1, 'Disque dur SSD 500GB', 75.00);
```

--------tp jcdb ----------

```
import java.sql.*;
import java.util.ArrayList;

public class ExoJDBC {
  private Connection con;

  public ExoJDBC(String url, String user, String password) throws SQLException {
    con = DriverManager.getConnection(url, user, password);
  }

  // Méthode pour récupérer les salles
  public ArrayList<String> getSalles() throws SQLException {
    ArrayList<String> liste = new ArrayList<>();
    Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
    ResultSet rs = stmt.executeQuery("SELECT * FROM Salle");

    while (rs.next()) {
      String salle = "nSalle=" + rs.getString("nSalle")
          + ", nomSalle=" + rs.getString("nomSalle")
          + ", nbPoste=" + rs.getInt("nbPoste")
          + ", indIP=" + rs.getString("indIP");
      liste.add(salle);
    }

    rs.close();
```

```
    stmt.close();
    return liste;
  }

  // Méthode pour ajouter une salle
  public void ajouterSalle(String nSalle, String nomSalle, int nbPoste, String indIP) throws SQLException {
    PreparedStatement ps = con.prepareStatement("INSERT INTO Salle(nSalle, nomSalle, nbPoste, indIP) VALUES (?, ?, ?, ?)");
    ps.setString(1, nSalle);
    ps.setString(2, nomSalle);
    ps.setInt(3, nbPoste);
    ps.setString(4, indIP);
    ps.executeUpdate();
    ps.close();
  }

  // Méthode pour supprimer une salle par son rang (curseur modifiable)
  public void deleteSalle(int rang) {
    try {
      Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
      ResultSet rs = stmt.executeQuery("SELECT * FROM Salle");

      if (rs.absolute(rang)) {
        rs.deleteRow();
        System.out.println("Salle supprimée.");
      } else {
        System.out.println("Rang invalide.");
```

```
        }

        rs.close();
        stmt.close();
      } catch (SQLException e) {
        if (e.getErrorCode() == 1451) {
          System.out.println("Erreur : la salle est référencée ailleurs (code 1451).");
        } else {
          e.printStackTrace();
        }
      }
    }

    public void close() throws SQLException {
      if (con != null) con.close();
    }
}
/* classe main */
```

---- je mets bien les appels à ma base avec le bon utilisateur et le bon mot de passe

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
      try {
        // Connexion
        ExoJDBC ej = new ExoJDBC("jdbc:mysql://localhost:3306/BDD", "ounissa", "jude");
```

```
// 1. Affiche les salles
System.out.println("Liste des salles :");
ArrayList<String> salles = ej.getSalles();
for (String s : salles) {
   System.out.println(s);
}

// 2. Ajoute une salle (test)
ej.ajouterSalle("S999", "SalleTest", 5, "192.168.1.0");

// 3. Affiche à nouveau
System.out.println("\nAprès insertion :");
salles = ej.getSalles();
for (String s : salles) {
   System.out.println(s);
}

// 4. Supprime la salle au dernier rang
ej.deleteSalle(salles.size());

// 5. Affiche encore une fois
System.out.println("\nAprès suppression :");
salles = ej.getSalles();
for (String s : salles) {
   System.out.println(s);
}

ej.close();
```

```
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

--------tp 7----------

---- Procédures stockées

```
-- Active l'affichage de résultats dans le client
DELIMITER //

CREATE PROCEDURE DerniereInstallationLogiciel()
BEGIN
   -- Variables pour stocker les résultats
   DECLARE vSalle VARCHAR(7);
   DECLARE vPoste VARCHAR(7);
   DECLARE vLog VARCHAR(20);
   DECLARE vDate DATE;

   -- Récupération de la dernière installation (dateIns la plus récente non nulle)
   SELECT s.nSalle, p.nPoste, l.nomLog, i.dateIns
   INTO vSalle, vPoste, vLog, vDate
   FROM Installer i
```

```
  JOIN Poste p ON i.nPoste = p.nPoste
  JOIN Salle s ON p.nSalle = s.nSalle
  JOIN Logiciel l ON i.nLog = l.nLog
  WHERE i.dateIns IS NOT NULL
  ORDER BY i.dateIns DESC
  LIMIT 1;

  -- Affichage du résultat 1
  SELECT 'Résultat 1 exo 1' AS titre
  UNION ALL
  SELECT CONCAT('Dernière installation en salle : ', vSalle);

  -- Affichage du résultat 2
  SELECT CONCAT('Résultat 2 exo 1 : poste ', vPoste,
          ', logiciel : ', vLog,
          ', en date du ', DATE_FORMAT(vDate, '%Y-%m-%d')) AS details;
END;
//

DELIMITER ;

---on l'appelle avec cette commande CALL DerniereInstallationLogiciel();


-------Avec variables de session

-- Affectation des variables de session
SET @salle := 's01';       -- numéro de salle à tester
```

```
SET @typePoste := 'TX';     -- type de poste à tester

-- Variables pour stocker les résultats
SET @nbPostes := 0;
SET @nbInstallations := 0;

-- Récupération du nombre de postes de ce type dans la salle
SELECT COUNT(*) INTO @nbPostes
FROM Poste
WHERE nSalle = @salle AND typePoste = @typePoste;

-- Récupération du nombre d'installations sur ces postes
SELECT COUNT(*) INTO @nbInstallations
FROM Installer i
JOIN Poste p ON i.nPoste = p.nPoste
WHERE p.nSalle = @salle AND p.typePoste = @typePoste;

-- Affichage du résultat
SELECT 'Résultat exo 2' AS titre
UNION ALL
SELECT CONCAT(@nbPostes, ' poste(s), installé(s) en salle ', @salle,
        ', ', @nbInstallations, ' installation(s) de type ', @typePoste, '.') AS resultat;


----- LA TRANSACTION

-- Variables de session pour les infos du logiciel
SET @nLog := 'log15';
```

```
SET @nomLog := 'MYSQL query';

SET @version := '1.4';

SET @typeLog := 'PCWS';

SET @prix := 95.00;


-- Variable pour le poste cible

SET @poste := 'p7';


-- Variables internes pour stocker les dates

SET @dateAch := CURRENT_TIMESTAMP;


-- Début de la transaction

START TRANSACTION;


-- Étape 1 : Insertion du logiciel

INSERT INTO Logiciel (nLog, nomLog, dateAch, version, typeLog, prix)

VALUES (@nLog, @nomLog, @dateAch, @version, @typeLog, @prix);


-- Affichage de confirmation

SELECT 'Logiciel inséré dans la base' AS message;

SELECT CONCAT('Date d\'achat : ', @dateAch) AS message;


-- Étape 2 : Pause de 5 secondes

SELECT 'Attente de 5 secondes avant installation...' AS message;

SELECT SLEEP(5);


-- Étape 3 : Calcul de la date d’installation et du délai

SET @dateIns := CURRENT_TIMESTAMP;
```

```
SET @delai := TIMESTAMPDIFF(SECOND, @dateAch, @dateIns);

-- Étape 4 : Insertion dans la table Installer
INSERT INTO Installer (nPoste, nLog, dateIns, delai)
VALUES (@poste, @nLog, @dateIns, @delai);

-- Étape 5 : Messages de traçage
SELECT CONCAT('Date d\'installation : ', @dateIns) AS message;
SELECT CONCAT('Logiciel installé sur le poste ', @poste) AS message;

-- Fin de la transaction
COMMIT;
```