Belcollin Jordan

E3FI-1I Esiee Paris

Tp base de données

TP 1 : Création de base de données

1.2 Création des tables

```
-- Script creParc.sql
-- Création de la base de données pour la gestion du parc informatique
-- Création de la table Types en premier car elle est référencée par d'autres tables
CREATE TABLE Types (
   typeLP VARCHAR(9) PRIMARY KEY,
   nomType VARCHAR(20) NOT NULL
);

-- Création de la table Segment
CREATE TABLE Segment (
   indIP VARCHAR(11) PRIMARY KEY,
   nomSegment VARCHAR(20) NOT NULL,
   etage TINYINT(1)
);

-- Création de la table Salle
CREATE TABLE Salle (
   nSalle VARCHAR(7) PRIMARY KEY,
   nomSalle VARCHAR(20) NOT NULL,
   nbPoste TINYINT(2),
   indIP VARCHAR(11),
   FOREIGN KEY (indIP) REFERENCES Segment(indIP)
);
```

```
-- Création de la table Poste
CREATE TABLE Poste (
   nPoste VARCHAR(7) PRIMARY KEY,
   nomPoste VARCHAR(20) NOT NULL,
   indIP VARCHAR(11),
   ad VARCHAR(3) CHECK (ad BETWEEN '0' AND '255'),
   typePoste VARCHAR(9),
   nSalle VARCHAR(7),
   FOREIGN KEY (indIP) REFERENCES Segment(indIP),
   FOREIGN KEY (nSalle) REFERENCES Salle(nSalle),
   FOREIGN KEY (typePoste) REFERENCES Types(typeLP)
);
-- Création de la table Logiciel
CREATE TABLE Logiciel (
   nLog VARCHAR(5) PRIMARY KEY,
   nomLog VARCHAR(20),
   dateAch DATETIME,
   version VARCHAR(7),
   typeLog VARCHAR(9),
   prix DECIMAL(6,2) CHECK (prix >= 0),
   FOREIGN KEY (typeLog) REFERENCES Types(typeLP)
);
-- Création de la table Installer
CREATE TABLE Installer (
   nPoste VARCHAR(7),
   nLog VARCHAR(5),
   numIns INTEGER(5),
   dateIns TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   delai SMALLINT,
   PRIMARY KEY (nPoste, nLog, numIns),
   FOREIGN KEY (nPoste) REFERENCES Poste(nPoste),
   FOREIGN KEY (nLog) REFERENCES Logiciel(nLog)
);
```

1.3 Structures des tables

```
-- Script descParc.sql
-- Description des tables de la base de données
DESCRIBE Types;
DESCRIBE Segment;
DESCRIBE Salle;
DESCRIBE Poste;
DESCRIBE Logiciel;
DESCRIBE Installer;
```

1.4 Destruction des tables

```
-- Script dropParc.sql
-- Suppression des tables de la base de données
DROP TABLE IF EXISTS Installer;
DROP TABLE IF EXISTS Logiciel;
DROP TABLE IF EXISTS Poste;
DROP TABLE IF EXISTS Salle;
DROP TABLE IF EXISTS Segment;
DROP TABLE IF EXISTS Types;
```

TP2 : Manipulation des données

2.1 Insertion de données

```
-- Script inParc.sql
-- Insertion des données dans les tables du parc informatique
-- 1. Insertion dans la table Types (doit être fait en premier car référencé par d'autres tables)
INSERT INTO Types (typeLP, nomType) VALUES
('TX', 'Terminal X-Window'),
('UNIX', 'Système Unix'),
('PCNT', 'PC Windows NT'),
```

```
('PCWS', 'PC Windows'),
('NC', 'Network Computer');

-- 2. Insertion dans la table Segment
INSERT INTO Segment (indIP, nomSegment, etage) VALUES
('130.120.80', 'Brin RDC', NULL),
('130.120.81', 'Brin 1er étage', NULL),
('130.120.82', 'Brin 2e étage', NULL);

-- 3. Insertion dans la table Salle
INSERT INTO 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');

-- 4. Insertion dans la table Poste
INSERT INTO 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', 'PCNT', 's21');
-- 5. Insertion dans la table Logiciel
INSERT INTO Logiciel (nLog, nomLog, dateAch, version, typeLog, prix) VALUES
('log1', 'Oracle 6', '1995-05-13', '6.2', 'UNIX', 3000.00),
('log2', 'Oracle 8', '1999-09-15', '8i', 'UNIX', 5600.00),
('log3', 'SQL Server', '1998-04-12', '7', 'PCNT', 2700.00),
('log4', 'Front Page', '1997-06-03', '5', 'PCWS', 500.00),
('log5', 'WinDev', '1997-05-12', '5', 'PCWS', 750.00),
('log6', 'SQL*Net', NULL, '2.0', 'UNIX', 500.00),
('log7', 'I.I.S.', '2002-04-12', '2', 'PCNT', 810.00),
('log8', 'DreamWeaver', '2003-09-21', '2.0', 'BeOS', 1400.00);
```

2.2 - Gestion d'une séquence

```
-- Suite du script inParc.sql
DROP TABLE IF EXISTS Installer;
CREATE TABLE Installer (
   nPoste VARCHAR(7),
   nLog VARCHAR(5),
   numIns INTEGER(5) AUTO_INCREMENT,
   dateIns TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   delai SMALLINT,
   PRIMARY KEY (nPoste, nLog, numIns),
   FOREIGN KEY (nPoste) REFERENCES Poste(nPoste),
   FOREIGN KEY (nLog) REFERENCES Logiciel(nLog)
) AUTO_INCREMENT=1;
```

-- Insertion des données dans la table Installer

INSERT INTO Installer (nPoste, nLog, dateIns, delai) VALUES

('p2', 'log1', '2003-05-15', NULL),

('p2', 'log2', '2003-09-17', NULL),

('p4', 'log5', NULL, NULL),

('p6', 'log6', '2003-05-20', NULL),

('p6', 'log1', '2003-05-20', NULL),

('p8', 'log2', '2003-05-19', NULL),

('p8', 'log6', '2003-05-20', NULL),

('p11', 'log3', '2003-04-20', NULL),

('p12', 'log4', '2003-04-20', NULL),

('p11', 'log7', '2003-04-20', NULL),

('p7', 'log7', '2002-04-01', NULL);

2.3 - Modification de données

-- Script modification.sql

-- 1. Mise à jour des étages dans la table Segment

UPDATE Segment SET etage = 0 WHERE indIP = '130.120.80';

UPDATE Segment SET etage = 1 WHERE indIP = '130.120.81';

UPDATE Segment SET etage = 2 WHERE indIP = '130.120.82';

-- 2. Diminution de 10% du prix des logiciels de type 'PCNT'

UPDATE Logiciel SET prix = prix * 0.9 WHERE typeLog = 'PCNT';

-- 3. Vérification des modifications

SELECT * FROM Segment;

SELECT nLog, typeLog, prix FROM Logiciel WHERE typeLog = 'PCNT';

TP3 : évolution d'un schéma

```
-- Script evolution.sql
-- Évolution du schéma de la base de données du parc informatique
-- Exercice 3.1 - Ajout de colonnes
ALTER TABLE Segment ADD COLUMN nbSalle TINYINT(2) DEFAULT 0;
ALTER TABLE Segment ADD COLUMN nbPoste TINYINT(2) DEFAULT 0;
ALTER TABLE Logiciel ADD COLUMN nbInstall TINYINT(2) DEFAULT 0;
ALTER TABLE Poste ADD COLUMN nbLog TINYINT(2) DEFAULT 0;

-- Vérification
DESCRIBE Segment;
DESCRIBE Logiciel;
DESCRIBE Poste;
SELECT * FROM Segment LIMIT 1;
SELECT * FROM Logiciel LIMIT 1;
SELECT * FROM Poste LIMIT 1;

-- Exercice 3.2 - Modification de colonnes
-- Augmentation de la taille de nomSalle
ALTER TABLE Salle MODIFY nomSalle VARCHAR(30);

-- Diminution de la taille de nomSegment à 15
ALTER TABLE Segment MODIFY nomSegment VARCHAR(15);

-- Tentative de diminution à 14 (cela échouera car certaines valeurs sont plus longues)
-- ALTER TABLE Segment MODIFY nomSegment VARCHAR(14);
-- Vérification
DESCRIBE Salle;
DESCRIBE Segment;
SELECT nomSalle FROM Salle LIMIT 1;
SELECT nomSegment FROM Segment LIMIT 1;
```

```
-- Exercice 3.3 - Ajout de contraintes

-- Contrainte pour éviter les installations multiples du même logiciel sur un poste

ALTER TABLE Installer ADD CONSTRAINT unique_installation

UNIQUE (nPoste, nLog);


-- Contraintes de clés étrangères manquantes

-- D'abord nous devons corriger les problèmes identifiés

-- 1. Problème avec les salles s22 et s23 qui référencent un segment inexistant (130.120.83)

-- Extraction des enregistrements problématiques

SELECT * FROM Salle WHERE indIP NOT IN (SELECT indIP FROM Segment);


-- Suppression des enregistrements problématiques

DELETE FROM Salle WHERE nSalle IN ('s22', 's23');


-- 2. Problème avec le logiciel log8 de type 'BeOS' non référencé

-- Extraction des enregistrements problématiques

SELECT * FROM Logiciel WHERE typeLog NOT IN (SELECT typeLP FROM Types);


-- Ajout du type manquant

INSERT INTO Types (typeLP, nomType) VALUES ('BeOS', 'Système Be');


-- Entre Salle et Segment

ALTER TABLE Salle ADD CONSTRAINT fk_salle_segment

FOREIGN KEY (indIP) REFERENCES Segment(indIP);


-- Entre Logiciel et Types

ALTER TABLE Logiciel ADD CONSTRAINT fk_logiciel_types

FOREIGN KEY (typeLog) REFERENCES Types(typeLP);

-- Vérification que les contraintes sont bien ajoutées

SELECT TABLE_NAME, COLUMN_NAME, CONSTRAINT_NAME, REFERENCED_TABLE_NAME,
REFERENCED_COLUMN_NAME

FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE

WHERE TABLE_SCHEMA = DATABASE() AND REFERENCED_TABLE_NAME IS NOT NULL;
```

```
-- Script dropParc.sql mis à jour
-- Suppression des tables en tenant compte des nouvelles contraintes
-- Ordre de suppression important pour respecter les contraintes de clés étrangères
DROP TABLE IF EXISTS Installer;
DROP TABLE IF EXISTS Logiciel;
DROP TABLE IF EXISTS Poste;
DROP TABLE IF EXISTS Salle;
DROP TABLE IF EXISTS Segment;
DROP TABLE IF EXISTS Types;
```

TP4 : recherche des données

4.1 - Création dynamique de tables

```
-- Script creationdynamique.sql
-- Création dynamique des tables softs et pCSells
-- Table softs à partir de la table Logiciel
CREATE TABLE softs AS
SELECT
  nomLog AS nomSoft,
  version,
  prix
FROM Logiciel;
-- Table pCSells à partir de la table Poste (uniquement PCWS et PCNT)
CREATE TABLE pCSells AS
SELECT
  nPoste AS np,
  nomPoste AS nomP,
  indIP AS seq,
  ad,
  typePoste AS typeP,
  nSalle AS salle
FROM Poste
```

```
WHERE typePoste IN ('PCWS', 'PCNT');
-- Vérification
SELECT * FROM softs;
SELECT * FROM pCSells;
```

4.2, 4.3, 4.4, 4.6 -Requetes et fonctions

```
-- Script requetes.sql
-- Requêtes sur la base de données du parc informatique
-- 1. Type du poste 'p6'
SELECT typePoste FROM Poste WHERE nPoste = 'p6';

-- 2. Noms des logiciels '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, nSalle
FROM Poste
WHERE typePoste IN ('UNIX', 'PCWS');

-- 4. Postes du segment '130.120.80' triés par numéros de salles décroissants
SELECT nomPoste, indIP, 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';
```

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

-- 7. Noms et adresses IP complètes des postes de type 'TX'
SELECT nomPoste, CONCAT(indIP, '.', ad) AS adresseIP
FROM Poste
WHERE typePoste = 'TX';

-- 8. Pour chaque poste, le nombre de logiciels installés
SELECT nPoste, COUNT(*) AS nbLogiciels
FROM Installer
GROUP BY nPoste;

-- 9. Pour chaque salle, le nombre de postes
SELECT nSalle, COUNT(*) AS nbPostes
FROM Poste
GROUP BY nSalle;

-- 10. Pour chaque logiciel, le nombre d'installations sur des postes différents
SELECT nLog, COUNT(DISTINCT nPoste) AS nbInstallations
FROM Installer
GROUP BY nLog;

-- 11. Moyenne des prix des logiciels 'UNIX'
SELECT AVG(prix) AS moyennePrixUNIX
FROM Logiciel
WHERE typeLog = 'UNIX';
```

```
-- 12. Plus récente date d'achat d'un logiciel
SELECT MAX(dateAch) AS dateAchatRecent
FROM Logiciel;

-- 13. Numéros des postes hébergeant 2 logiciels
SELECT nPoste
FROM Installer
GROUP BY nPoste
HAVING COUNT(*) = 2;

-- 14. Nombre de postes hébergeant 2 logiciels
SELECT COUNT(*) AS nbPostes2Logiciels
FROM (
  SELECT nPoste
  FROM Installer
  GROUP BY nPoste
  HAVING COUNT(*) = 2
) AS postes2Logiciels;

-- 15. Types de postes non recensés dans le parc informatique
SELECT typeLP
FROM Types
WHERE typeLP NOT IN (SELECT DISTINCT typePoste FROM Poste WHERE typePoste IS NOT NULL);

-- 16. Types existant à la fois comme types de postes et de logiciels
SELECT DISTINCT typePoste
FROM Poste
WHERE typePoste IN (SELECT typeLog FROM Logiciel);
```

-- 17. Types de postes de travail n'étant pas des types de logiciels

SELECT DISTINCT typePoste

FROM Poste

WHERE typePoste NOT IN (SELECT typeLog FROM Logiciel WHERE typeLog IS NOT NULL)

AND typePoste IS NOT NULL;

-- 18. Adresses IP complètes des postes qui hébergent le logiciel 'log6' (version procédurale)

SELECT CONCAT(p.indIP, '.', p.ad) AS adresseIP

FROM Poste p, Installer i

WHERE p.nPoste = i.nPoste AND i.nLog = 'log6';

-- 19. Adresses IP complètes des postes qui hébergent le logiciel de nom 'Oracle 8' (version procédurale)

SELECT CONCAT(p.indIP, '.', p.ad) AS adresseIP

FROM Poste p, Installer i, Logiciel l

WHERE p.nPoste = i.nPoste AND i.nLog = l.nLog AND l.nomLog = 'Oracle 8';

-- 20. Noms des segments possédant exactement trois postes de travail de type 'TX' (version procédurale)

SELECT s.nomSegment

FROM Segment s, Poste p

WHERE s.indIP = p.indIP AND p.typePoste = 'TX'

GROUP BY s.indIP, s.nomSegment

HAVING COUNT(*) = 3;

-- 21. Noms des salles où l'on peut trouver au moins un poste hébergeant le logiciel 'Oracle 6' (version procédurale)

```
SELECT DISTINCT sa.nomSalle
FROM Salle sa, Poste p, Installer i, Logiciel l
WHERE sa.nSalle = p.nSalle AND p.nPoste = i.nPoste AND i.nLog = l.nLog AND l.nomLog = 'Oracle 6';

-- 22. Nom du logiciel acheté le plus récent
SELECT nomLog
FROM Logiciel
WHERE dateAch = (SELECT MAX(dateAch) FROM Logiciel);

-- 23. Requête 18 avec jointure relationnelle
SELECT CONCAT(p.indIP, '.', p.ad) AS adresseIP
FROM Poste p
JOIN Installer i ON p.nPoste = i.nPoste
WHERE i.nLog = 'log6';

-- 24. Requête 19 avec jointure relationnelle
SELECT CONCAT(p.indIP, '.', p.ad) AS adresseIP
FROM Poste p
JOIN Installer i ON p.nPoste = i.nPoste
JOIN Logiciel l ON i.nLog = l.nLog
WHERE l.nomLog = 'Oracle 8';

-- 25. Requête 20 avec jointure relationnelle
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;
```

```
-- 26. Requête 21 avec jointure relationnelle
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';

-- 27. Installations complètes triées
SELECT s.nomSegment, sa.nomSalle, CONCAT(p.indIP, '.', p.ad) AS adresseIP, l.nomLog, i.dateIns
FROM Segment s
JOIN Salle sa ON s.indIP = sa.indIP
JOIN Poste p ON sa.nSalle = p.nSalle
JOIN Installer i ON p.nPoste = i.nPoste
JOIN Logiciel l ON i.nLog = l.nLog
ORDER BY s.nomSegment, sa.nomSalle, adresseIP;

-- 28. Requête 18 avec jointure SQL2
SELECT CONCAT(p.indIP, '.', p.ad) AS adresseIP
FROM Poste p
INNER JOIN Installer i USING (nPoste)
WHERE i.nLog = 'log6';
-- 29. Requête 19 avec jointure SQL2
SELECT CONCAT(p.indIP, '.', p.ad) AS adresseIP
FROM Poste p
INNER JOIN Installer i USING (nPoste)
INNER JOIN Logiciel l USING (nLog)
WHERE l.nomLog = 'Oracle 8';
```

```sql
-- 30. Requête 20 avec jointure SQL2
SELECT s.nomSegment
FROM Segment s
INNER JOIN Poste p USING (indIP)
WHERE p.typePoste = 'TX'
GROUP BY s.indIP, s.nomSegment
HAVING COUNT(*) = 3;

-- 31. Requête 21 avec jointure SQL2
SELECT DISTINCT sa.nomSalle
FROM Salle sa
INNER JOIN Poste p USING (nSalle)
INNER JOIN Installer i USING (nPoste)
INNER JOIN Logiciel l USING (nLog)
WHERE l.nomLog = 'Oracle 6';

-- 32. Noms des postes ayant au moins un logiciel commun au poste 'p6'
SELECT DISTINCT p.nomPoste
FROM Poste p
JOIN Installer i ON p.nPoste = i.nPoste
WHERE i.nLog IN (SELECT nLog FROM Installer WHERE nPoste = 'p6') AND p.nPoste !=
'p6';
-- 33. Noms des postes ayant les mêmes logiciels que le poste 'p6' (division inexacte)
SELECT p.nomPoste
FROM Poste p
WHERE NOT EXISTS (
  SELECT nLog FROM Installer WHERE nPoste = 'p6'
  EXCEPT
  SELECT nLog FROM Installer WHERE nPoste = p.nPoste
) AND p.nPoste != 'p6';
```

```
-- 34. Noms des postes ayant exactement les mêmes logiciels que le poste 'p2' (division exacte)
SELECT p.nomPoste
FROM Poste p
WHERE NOT EXISTS (
  SELECT nLog FROM Installer WHERE nPoste = 'p2'
  EXCEPT
  SELECT nLog FROM Installer WHERE nPoste = p.nPoste
)
AND NOT EXISTS (
  SELECT nLog FROM Installer WHERE nPoste = p.nPoste
  EXCEPT
  SELECT nLog FROM Installer WHERE nPoste = 'p2'
)
AND p.nPoste != 'p2';
```

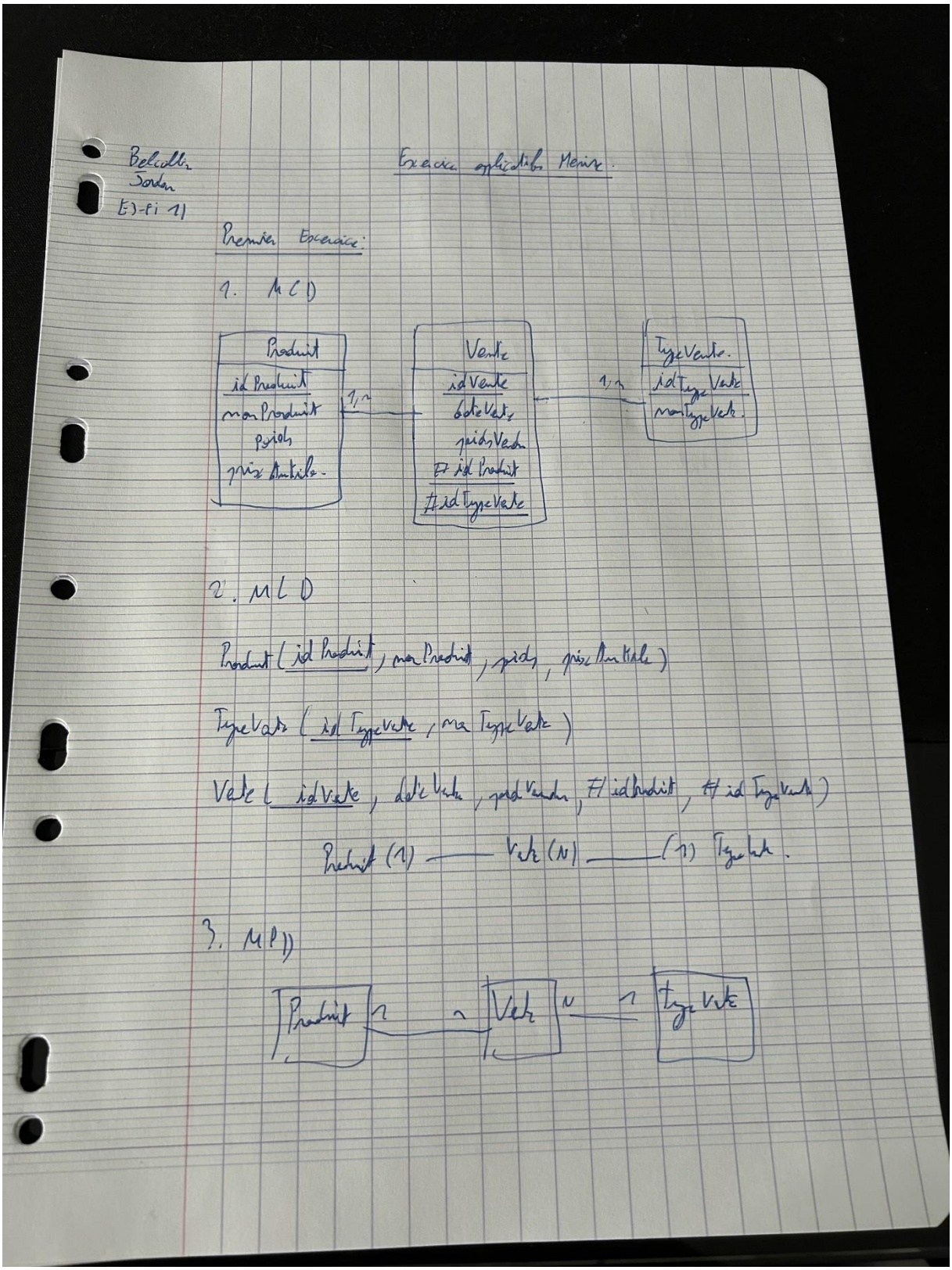
4.5 - Modifications synchronisées

```
-- Script modificationsynchronisees.sql
-- Modifications synchronisées de la base de données
-- Ajout des nouvelles installations
INSERT INTO Installer (nPoste, nLog, dateIns, delai) VALUES
('p2', 'log6', SYSDATE(), NULL),
('p8', 'log1', SYSDATE(), NULL),
('p10', 'log1', SYSDATE(), NULL);
-- Mise à jour synchronisée des colonnes ajoutées
-- Nombre de salles par segment
UPDATE Segment s
SET nbSalle = (
  SELECT COUNT(*)
  FROM Salle sa
  WHERE sa.indIP = s.indIP
```

```
);

-- Nombre de postes par segment
UPDATE Segment s
SET nbPoste = (
  SELECT COUNT(*)
  FROM Poste p
  WHERE p.indIP = s.indIP
);
-- Nombre d'installations par logiciel
UPDATE Logiciel l
SET nbInstall = (
  SELECT COUNT(*)
  FROM Installer i
  WHERE i.nLog = l.nLog
);
-- Nombre de logiciels par poste
UPDATE Poste p
SET nbLog = (
  SELECT COUNT(*)
  FROM Installer i
  WHERE i.nPoste = p.nPoste
);
-- Vérification
SELECT * FROM Segment;
SELECT * FROM Logiciel;
SELECT * FROM Poste;
```

Tp 5 : Exercices Applicatifs sur Merise



4. Exemple concret tables

Type Vente : 1, Animaux
2, Légumes,
3, Fruits

Produits : 1, 'Lapin', 2.5, 10.0
2, 'Poule', 1.8, 8.0
3, 'Pomme de Terre', 5.0, 1.5

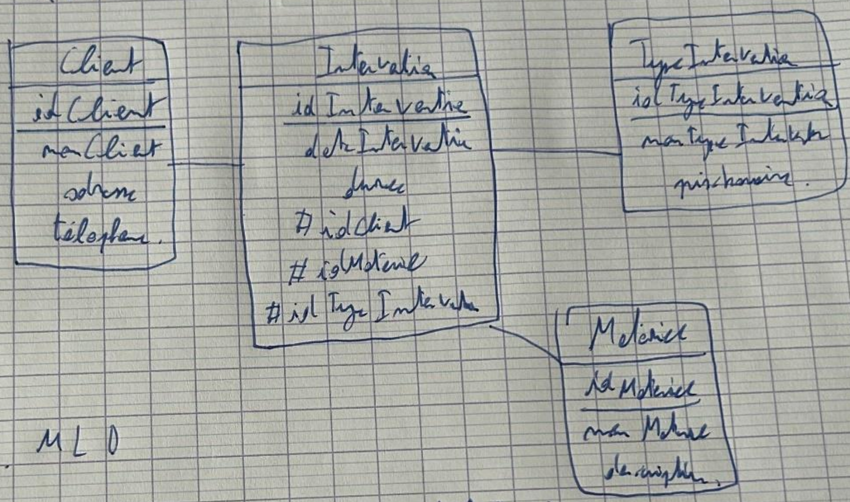
Vente : 1, '2025-10-01', 2.5, 1, 1.
2, '2025-10-02', 3.0, 3, 2

Belcalkin
Jordan
E3-FI 1L

Exercice applicatif Merise

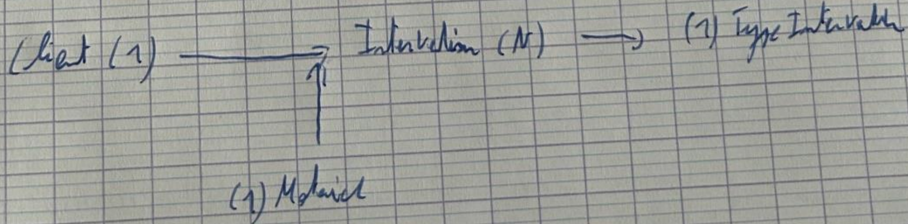
Deuxième Exercice :

1. MCD

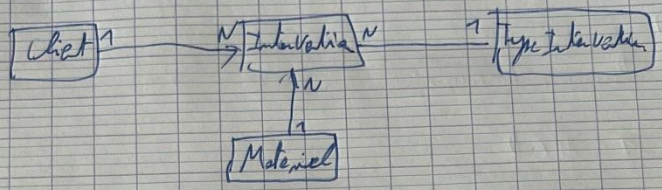


2. MLD

Client (idClient, nomClient, adresse, téléphone)
Materiel (idMateriel, nomMateriel, description)
TypeIntervention (idTypeIntervention, nomTypeIntervention, prixHoraire)
Intervention (idIntervention, dateIntervention, durée, # idClient, # idMateriel, # idTypeIntervention)



3. MPD



TP 6 : Java-JDBC

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

```
public class ExoJDBC {
  private Connection connection;

  // Constructeur
  public ExoJDBC(String driver, String url, String login, String password)
      throws ClassNotFoundException, SQLException {
    // Chargement du driver JDBC
    Class.forName(driver);
    // Établissement de la connexion
    this.connection = DriverManager.getConnection(url, login, password);
  }

  // Méthode pour fermer la connexion
  public void close() throws SQLException {
    if (connection != null) {
      connection.close();
    }
  }
  public ArrayList<String> getSalles() throws SQLException {
  ArrayList<String> salles = new ArrayList<>();
  String query = "SELECT nsalle, nomsalle, nbPoste, indIP FROM Salle";

  try (Statement stmt = connection.createStatement();
     ResultSet rs = stmt.executeQuery(query)) {

    // Formatage des résultats
    while (rs.next()) {
      String ligne = String.format("%-10s %-20s %-8d %-15s",
          rs.getString("nsalle"),
```

```
                rs.getString("nomsalle"),
                rs.getInt("nbPoste"),
                rs.getString("indIP"));
            salles.add(ligne);
        }
    }
    return salles;
}

}

public static void main(String[] args) {
    // Paramètres de connexion
    String driver = "com.mysql.cj.jdbc.Driver";
    String url = "jdbc:mysql://localhost:3306/parc_informatique";
    String login = "votre_login";
    String password = "votre_mot_de_passe";

    try (ExoJDBC exo = new ExoJDBC(driver, url, login, password)) {
        // Récupération des salles
        ArrayList<String> salles = exo.getSalles();

        // Affichage des résultats
        System.out.println("Liste des salles:");
        System.out.println("-------------------------------------------");
        System.out.printf("%-10s %-20s %-8s %-15s%n",
                "nsalle", "nomsalle", "nbPoste", "indIP");
        System.out.println("-------------------------------------------");
```

```
        for (String salle : salles) {
            System.out.println(salle);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void deleteSalle(int rang) throws SQLException {
    String query = "SELECT nsalle, nomsalle, nbPoste, indIP FROM Salle";

    try (Statement stmt = connection.createStatement(
            ResultSet.TYPE_SCROLL_SENSITIVE,
            ResultSet.CONCUR_UPDATABLE);
         ResultSet rs = stmt.executeQuery(query)) {

        // Se positionner sur la ligne à supprimer
        if (rs.absolute(rang)) {
            try {
                rs.deleteRow();
                System.out.println("Salle supprimée avec succès.");
            } catch (SQLException e) {
                if (e.getErrorCode() == 1451) {
                    System.out.println("Erreur 1451: Impossible de supprimer " +
                            "car la salle est référencée par d'autres tables.");
                } else {
                    throw e;
                }
```

```
        }
      } else {
        System.out.println("Aucune salle trouvée au rang " + rang);
      }
    }
}

public void actionPerformed(ActionEvent evt) {
   if (evt.getSource() == boutonConnection) {
      try {
        // Initialisation de la connexion
        ExoJDBC exo = new ExoJDBC(
          nomDriver.getText(),
          urlConnection.getText(),
          nomLogin.getText(),
          motPasse.getText());

        // Récupération des salles
        ArrayList<String> salles = exo.getSalles();
        resultatRequete.removeAll();

        // Affichage des résultats
        resultatRequete.add("Liste des salles:");
        resultatRequete.add("---------------------------------");
        resultatRequete.add(String.format("%-10s %-20s %-8s %-15s",
          "nsalle", "nomsalle", "nbPoste", "indIP"));
        resultatRequete.add("---------------------------------");

        for (String salle : salles) {
```

```
            resultatRequete.add(salle);
        }

        exo.close();
    } catch (Exception e) {
        resultatRequete.add("Erreur: " + e.getMessage());
    }
} else if (evt.getSource() == boutonExecuter) {
    // Exécution d'une requête SQL personnalisée
    try (ExoJDBC exo = new ExoJDBC(
            nomDriver.getText(),
            urlConnection.getText(),
            nomLogin.getText(),
            motPasse.getText());
         Statement stmt = exo.getConnection().createStatement();
         ResultSet rs = stmt.executeQuery(requeteSQL.getText())) {

        resultatRequete.removeAll();
        ResultSetMetaData meta = rs.getMetaData();
        int colCount = meta.getColumnCount();

        // Affichage des noms de colonnes
        StringBuilder header = new StringBuilder();
        for (int i = 1; i <= colCount; i++) {
            header.append(String.format("%-20s", meta.getColumnName(i)));
        }
        resultatRequete.add(header.toString());

        // Affichage des données
```

```
        while (rs.next()) {
            StringBuilder row = new StringBuilder();
            for (int i = 1; i <= colCount; i++) {
                row.append(String.format("%-20s", rs.getString(i)));
            }
            resultatRequete.add(row.toString());
        }
    } catch (Exception e) {
        resultatRequete.add("Erreur: " + e.getMessage());
    }
  }
}
```

TP 7 Les procédures stockées

6.1 - Extraction des données de la dernière installation

```
DELIMITER //
CREATE PROCEDURE derniere_installation()
```

```
BEGIN
  DECLARE v_nsalle VARCHAR(7);
  DECLARE v_nposte VARCHAR(7);
  DECLARE v_nomlog VARCHAR(20);
  DECLARE v_dateins TIMESTAMP;

  -- Récupération des informations de la dernière installation
  SELECT s.nSalle, i.nPoste, l.nomLog, i.dateIns
  INTO v_nsalle, v_nposte, v_nomlog, v_dateins
  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
  ORDER BY i.dateIns DESC
  LIMIT 1;

  -- Affichage des résultats
  SELECT CONCAT('Derniere installation en salle : ', v_nsalle) AS 'Resultat 1 exo 1';
  SELECT CONCAT('Poste : ', v_nposte, ' Logiciel : ', v_nomlog,
        ' en date du ', DATE_FORMAT(v_dateins, '%Y-%m-%d %H:%i:%s')) AS 'Resultat 2 exo 1';
END //
DELIMITER ;
```

6.2 - Variables de session avec comptage

```
DELIMITER //
CREATE PROCEDURE stats_poste_installation(
  IN p_nsalle VARCHAR(7),
  IN p_typeposte VARCHAR(9)
)
```

```
BEGIN
  DECLARE v_nbpostes INT;
  DECLARE v_nbinstallations INT;

  -- Compter le nombre de postes dans la salle avec le type spécifié
  SELECT COUNT(*) INTO v_nbpostes
  FROM Poste
  WHERE nSalle = p_nsalle AND typePoste = p_typeposte;

  -- Compter le nombre d'installations pour ces postes
  SELECT COUNT(*) INTO v_nbinstallations
  FROM Installer i
  JOIN Poste p ON i.nPoste = p.nPoste
  WHERE p.nSalle = p_nsalle AND p.typePoste = p_typeposte;

  -- Afficher le résultat formaté
  SELECT CONCAT(v_nbpostes, ' poste(s) installe(s) en salle ', p_nsalle,
          ', ', v_nbinstallations, ' installation(s) de type ', p_typeposte) AS 'Resultat exo2';
END //

DELIMITER ;
```