## Projektaufgabe 2

# Phase 1 – Legen der Basis (2.5 P)

### Datenmanagement jenseits von Relationen

Gruppen Nummer (e.g. A1, B5, B3)
Weilert Alexander, 12119653
Jovanovic Dragana, StudentID2
May 12, 2024

Dieses Reporting Template dient der Vorbereitung der Abgabe von Phase 1.

#### Datengenerator für Matrizen mit Sparsity (0.5 Punkte)

Zeigen Sie den Code der Funktion generate() als Listing oder Screenshot. Gehen Sie (kurz) auf die wesentlichen Aspekte ein.

```
[Your answer goes here ...]
 public void generate(int 1, double sparsity) {
    try (Statement statement = this.connection.createStatement()) {
        statement.execute("DROP VIEW IF EXISTS C");
        statement.execute("DROP TABLE IF EXISTS A, B");
        // Create Table
        statement.execute("CREATE TABLE A (i int, j int, val int)");
        statement.execute("CREATE TABLE B (i int, j int, val int)");
        int[][] matrixA = generateMatrixA(1, sparsity);
        int[][] matrixB = generateMatrixB(1, sparsity);
        insertMatrix("A", matrixA);
        insertMatrix("B", matrixB);
        ansatzO(matrixA, matrixB); // Matrix Calculator per Algorithm
                                   // Matrix Calculator per Select
        ansatz1();
    } catch (SQLException e) {
        throw new RuntimeException(e);
}
public int[][] generateMatrixA(int 1, double sparsity) {
    Random random = new Random();
    int[][] matrixA = new int[1-1][1];
    System.out.println("--- Matrix A ---");
    for (int i = 0; i < ( 1 - 1 ); i++) {
        for (int j = 0; j < 1; j++) {
            if (random.nextDouble() > sparsity) {
```

```
matrixA[i][j] = random.nextInt(1, 11); // Random value between 0
            } else {
                matrixA[i][j] = 0;
            System.out.print(matrixA[i][j] + " ");
        System.out.println();
    return matrixA;
}
public int[][] generateMatrixB(int 1, double sparsity) {
    Random random = new Random();
    int[][] matrixB = new int[l][l-1];
    System.out.println("--- Matrix B ---");
    for (int i = 0; i < 1; i++) {
        for (int j = 0; j < (1 - 1); j++) {
            if (random.nextDouble() > sparsity) {
                matrixB[i][j] = random.nextInt(1, 11); // Random value between 0 a
            } else {
                matrixB[i][j] = 0;
            System.out.print(matrixB[i][j] + " ");
        System.out.println();
    return matrixB;
}
public void insertMatrix(String tableName, int[][] matrix) {
    try (Statement statement = this.connection.createStatement()) {
        StringBuilder insertQuery = new StringBuilder("INSERT INTO " + tableName
        for (int i = 0; i < matrix.length; i++) {</pre>
            for (int j = 0; j < matrix[i].length; j++) {</pre>
                if (matrix[i][j] != 0) {
                     insertQuery.append("(").append(i+1).append(",").append(j+1).a
                }
            }
        }
        insertQuery.deleteCharAt(insertQuery.length() - 1);
        statement.executeUpdate(insertQuery.toString());
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

## Import der Matrizen in das DBMS (0.5 Punkte)

Geben Sie das Create Table Statement für Matrix A an.

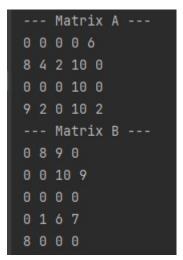
```
CREATE TABLE A (i int, j int, val int)
```

Für die Übung bereiten Sie eine Demo des Datenimports vor.

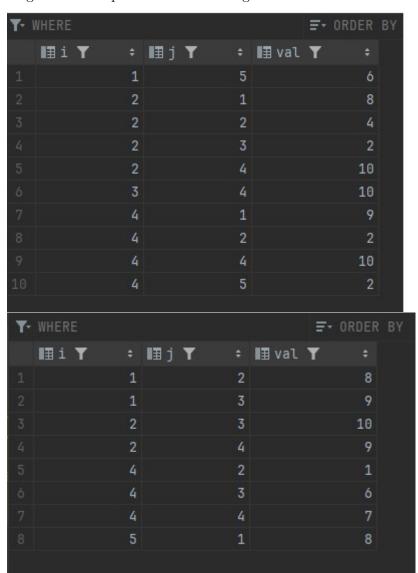
[Your answer goes here ...]

## Wahl des Toy Beispiels (0.5 Punkte)

Geben Sie Matrix A und B als 2D Array an.



Zeigen Sie die äquivalente Darstellung der Matrix A und B in der Datenbank.



#### Implementierung von Ansatz 0 (0.5 Punkte)

Zeigen Sie den Code der Matrixmultiplikation als Listing oder Screenshot. Erläutern Sie (kurz), welche Laufzeit ihr Algorithmus hat und warum das Kriterium keinen Algorithmus mit sub-kubischer Laufzeit zu wählen erfüllt ist.

```
public void ansatz0(int[][] matrixA, int[][] matrixB) {
    try (Statement statement = this.connection.createStatement()) {
        int[][] result = new int[matrixA.length][matrixB[0].length];
        for (int i = 0; i < matrixA.length; i++) {</pre>
            for (int j = 0; j < matrixB[0].length; j++) {</pre>
                 for (int k = 0; k < matrixA[0].length; k++) {</pre>
                     result[i][j] += matrixA[i][k] * matrixB[k][j];
                 }
            }
        System.out.println("--- Matrix Calculator ---");
        for (int i = 0; i < result.length; i++) {</pre>
            for (int j = 0; j < result.length; <math>j++) {
                 System.out.print(result[i][j] + " ");
            System.out.println();
        }
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

#### Implementierung von Ansatz 1 (0.5 Punkte)

Berechnen Sie von Hand das Ergebnis  $C = A \times B$  für ihr Toy Beispiel und geben Sie es nachfolgend an.

Zeigen Sie, dass Ihr System C korrekt berechnet (z.B. als Screenshot)

## Zeitmamagement

Benötigte Zeit pro Person (nur Phase 1):

Alexander Weilert: 5h Dragana Jovanovic: 5h

# References

Important: Reference your information sources!
Remove this section if you use footnotes to reference your information sources.