

## 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 l, 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(l, sparsity);
        int[][] matrixB = generateMatrixB(l, sparsity);
        insertMatrix("A", matrixA);
        insertMatrix("B", matrixB);

        ansatz0(matrixA, matrixB); // Matrix Calculator per Algorithm
        ansatz1();                // Matrix Calculator per Select

    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}

public int[][] generateMatrixA(int l, double sparsity) {
    Random random = new Random();
    int[][] matrixA = new int[l-1][l];
    System.out.println("--- Matrix A ---");
    for (int i = 0; i < (l - 1); i++) {
        for (int j = 0; j < l; 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 l, double sparsity) {
    Random random = new Random();
    int[][] matrixB = new int[l][l-1];
    System.out.println("--- Matrix B ---");
    for (int i = 0; i < l; i++) {
        for (int j = 0; j < (l - 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++) {
            for (int j = 0; j < matrix[i].length; j++) {
                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.

```
--- Matrix A ---
0 0 0 0 6
8 4 2 10 0
0 0 0 10 0
9 2 0 10 2
--- Matrix B ---
0 8 9 0
0 0 10 9
0 0 0 0
0 1 6 7
8 0 0 0
```

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

	WHERE		ORDER BY
	i	j	val
1	1	5	6
2	2	1	8
3	2	2	4
4	2	3	2
5	2	4	10
6	3	4	10
7	4	1	9
8	4	2	2
9	4	4	10
10	4	5	2

	WHERE		ORDER BY
	i	j	val
1	1	2	8
2	1	3	9
3	2	3	10
4	2	4	9
5	4	2	1
6	4	3	6
7	4	4	7
8	5	1	8

### 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++) {
            for (int j = 0; j < matrixB[0].length; j++) {
                for (int k = 0; k < matrixA[0].length; k++) {
                    result[i][j] += matrixA[i][k] * matrixB[k][j];
                }
            }
        }
        System.out.println("--- Matrix Calculator ---");
        for (int i = 0; i < result.length; i++) {
            for (int j = 0; j < result[0].length; 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.

```
public void ansatz1() {
    try (Statement statement = this.connection.createStatement()) {
        statement.execute("CREATE VIEW C AS " +
            "SELECT a.i, b.j, SUM(A.val * B.val) " +
            "FROM a, b " +
            "WHERE a.j = b.i " +
            "GROUP BY a.i, b.j");
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

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

### Zeitmanagement

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.

---