

Task 1 Connected Components

```
DROP PROCEDURE IF EXISTS ConnectedComponents;
GO

CREATE PROCEDURE ConnectedComponents
AS
BEGIN
    IF OBJECT_ID('temp_edges') IS NOT NULL DROP TABLE temp_edges;
    SELECT paperID, citedPaperID INTO temp_edges FROM edges
    UNION
    SELECT citedPaperID, paperID FROM edges;

    IF OBJECT_ID('components') IS NOT NULL DROP TABLE components;
    CREATE TABLE components (
        paperID INT,
        compID INT
    );

    DECLARE @compID INT = 0;

    DECLARE curs CURSOR FOR SELECT paperID FROM nodes;
    DECLARE @start INT;

    OPEN curs;
    FETCH NEXT FROM curs INTO @start;
    WHILE @@FETCH_STATUS = 0
    BEGIN
        IF NOT EXISTS (SELECT 1 FROM components WHERE paperID = @start)
        BEGIN
            SET @compID = @compID + 1;

            IF OBJECT_ID('queue') IS NOT NULL DROP TABLE queue;
            CREATE TABLE queue (paperID INT PRIMARY KEY);

            INSERT INTO queue VALUES (@start);

            WHILE EXISTS (SELECT 1 FROM queue)
            BEGIN
                INSERT INTO components (paperID, compID)
                SELECT q.paperID, @compID
                FROM queue q
                WHERE NOT EXISTS (SELECT 1 FROM components c WHERE c.paperID =
q.paperID);

                DELETE q
                FROM queue q
                JOIN components c ON q.paperID = c.paperID;

                INSERT INTO queue
                SELECT DISTINCT e.citedPaperID
                FROM temp_edges e
                JOIN components c ON e.paperID = c.paperID AND c.compID =
@compID
                WHERE NOT EXISTS (SELECT 1 FROM components cc WHERE cc.paperID
= e.citedPaperID);
            END
        END
    END
```

```

        END
    END

    FETCH NEXT FROM curs INTO @start;
END
CLOSE curs;
DEALLOCATE curs;

SELECT c.compID, n.paperID, n.paperTitle
FROM components c
JOIN nodes n ON c.paperID = n.paperID
WHERE c.compID IN (
    SELECT compID FROM components GROUP BY compID HAVING COUNT(*) BETWEEN
5 AND 10
)
ORDER BY c.compID, n.paperID;
END;
GO

EXEC ConnectedComponents;

```

The screenshot shows a database IDE with a SQL script editor and an output window. The script defines a procedure named `ConnectedComponents` that drops a table `temp_edges` if it exists, then inserts data from the `edges` table into it. The output window displays the results of the procedure, showing a table with columns `compID`, `paperID`, and `paperTitle`.

SQL Script:

```

1 DROP PROCEDURE IF EXISTS ConnectedComponents;
2 GO
3
4 CREATE PROCEDURE ConnectedComponents
5 AS
6 BEGIN
7     IF OBJECT_ID('temp_edges') IS NOT NULL DROP TABLE temp_edges;
8     SELECT paperID, citedPaperID INTO temp_edges FROM edges
9     UNION
10    SELECT paperID, citedPaperID, citedPaperID, paperID FROM edges;

```

Output Results:

compID	paperID	paperTitle
1	3	8110 Understanding Skyrmions using Rational Maps
2	3	12215 Solitonic fullerene structures in light atomic nuclei
3	3	206160 Skyrmion Monopoles
4	3	210310 Homotopy of Rational Maps and the Quantization of Skyrmions
5	3	9984160 Spherically Symmetric Solutions of the SU(N) Skyrme Models
6	8	9611150 Dimensional Renormalization in ϕ^4 theory: ladders and rainbows
7	8	9612010 Weight Systems from Feynman Diagrams
8	8	9712140 Non-zeta knots in the renormalization of the Wess-Zumino model?
9	8	9805025 A dilogarithmic 3-dimensional Ising tetrahedron
10	8	9807125 How useful can knot and number theory be for loop calculations?
11	10	9507110 Calogero-Sutherland model from excitations of Chern-Simons vortices
12	10	9611185 A Nonrelativistic Chiral Soliton in One Dimension
13	10	9706080 Moving Frames Hierarchical and RG Theory
14	10	9709075 Chiral solitons from reduction of Chern-Simons gauged

Task 2 PageRank

```
DROP PROCEDURE IF EXISTS ComputePR;
GO

CREATE PROCEDURE ComputePR
AS
BEGIN
    DECLARE @d FLOAT = 0.85;
    DECLARE @delta FLOAT = 1.0;
    DECLARE @n INT;

    SELECT @n = COUNT(*) FROM nodes;

    IF OBJECT_ID('pagerank') IS NOT NULL DROP TABLE pagerank;
    CREATE TABLE pagerank (paperID INT PRIMARY KEY, rank FLOAT);

    INSERT INTO pagerank
    SELECT paperID, 1.0/@n FROM nodes;

    IF OBJECT_ID('outdeg') IS NOT NULL DROP TABLE outdeg;
    SELECT paperID, COUNT(*) AS outdeg INTO outdeg FROM edges GROUP BY
paperID;

    WHILE @delta > 0.01
    BEGIN
        IF OBJECT_ID('newrank') IS NOT NULL DROP TABLE newrank;
        CREATE TABLE newrank (paperID INT PRIMARY KEY, rank FLOAT);

        INSERT INTO newrank
        SELECT n.paperID,
            (1.0 - @d)/@n +
            @d * (
                ISNULL(
                    SELECT SUM(pr.rank / od.outdeg)
                    FROM edges e
                    JOIN pagerank pr ON e.paperID = pr.paperID
                    JOIN outdeg od ON e.paperID = od.paperID
                    WHERE e.citedPaperID = n.paperID
                ), 0)
        FROM nodes n;

        DECLARE @sinkRank FLOAT = (
            SELECT SUM(pr.rank)
            FROM pagerank pr
            WHERE pr.paperID NOT IN (SELECT paperID FROM outdeg)
        );

        UPDATE newrank
        SET rank = rank + @d * @sinkRank / @n;

        SELECT @delta = SUM(ABS(n.rank - p.rank))
        FROM newrank n
        JOIN pagerank p ON n.paperID = p.paperID;

        DELETE FROM pagerank;
```

```

        INSERT INTO pagerank SELECT * FROM newrank;
    END;

    PRINT 'Top 10 Papers by PageRank:';
    SELECT TOP 10 n.paperID, n.paperTitle, p.rank
    FROM pagerank p
    JOIN nodes n ON p.paperID = n.paperID
    ORDER BY p.rank DESC;
END;
GO
EXEC ComputePR;

```

The screenshot shows a SQL development environment with the following components:

- Database Explorer:** Shows the database structure for `@classdb.ad.rice.edu`, including `DB_SQ275` (0 of 13), `master` (0 of 13), and `Server Objects`.
- Code Editor:** Displays the SQL script `sg275_PageRank.sql` with the following content:


```

1 DROP PROCEDURE IF EXISTS ComputePR;
2 GO
3
4 CREATE PROCEDURE ComputePR
5 AS
6 BEGIN
7     DECLARE @d FLOAT = 0.85;
8     DECLARE @delta FLOAT = 1.0;
9     DECLARE @n INT;
10

```
- Output Window:** Shows the results of the stored procedure execution. The output is a table with 10 rows, displaying the top 10 papers by PageRank. The columns are `paperID`, `paperTitle`, and `rank`.

paperID	paperTitle	rank
9504090	Massless Black Holes and Conifolds in String Theory	0.014726301489589675
9510135	Bound States Of Strings And p-Branes	0.01444560734542386
9711200	The Large N Limit of Superconformal Field Theories and Supergravity	0.01364692185545901
9802150	Anti De Sitter Space And Holography	0.009697437359075944
208020	Open strings and their symmetry groups	0.0086311043505559
9602065	D--branes and Spinning Black Holes	0.007717399363362192
9305185	Duality Symmetries of 4D Heterotic Strings	0.007549428739849736
9611050	TASI Lectures on D-Branes	0.007129032554566822
9581030	Strong/Weak Coupling Duality from the Dual String	0.005815174142168702
9602135	Entropy and Temperature of Black 3-Branes	0.005415907562197611

The status bar at the bottom indicates the file path: `Users > gkrs_17 > Documents > COMP_543_ToolsAndModelsForDS > sg275_Assignment2 > sg275_PageRank.sql`. It also shows the file size (9:20 (1834 chars, 64 line breaks)), encoding (UTF-8), and tab settings (4 spaces).